

repository.ub.ac.id

# **PERANCANGAN SISTEM PENGAMANAN DATA TRANSAKSI PADA *DATABASE* TERDISTRIBUSI MENGGUNAKAN METODE *HASHING***

## **SKRIPSI**

Untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun oleh:  
Agung Pambudi  
145150200111062



PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2018

## PENGESAHAN

PERANCANGAN SISTEM PENGAMANAN DATA TRANSAKSI PADA DATABASE  
TERDISTRIBUSI MENGGUNAKAN METODE *HASHING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun Oleh:  
Agung Pambudi  
NIM: 145150200111062

Skripsi ini telah diuji dan dinyatakan lulus pada  
1 Agustus 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II

Ari Kusyanti, S.T, M.Sc  
NIK: 2011028312282001

Mahendra Data, S.Kom., M.Kom  
NIK: 2015038611171001

Mengetahui  
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D  
NIP: 197105182003121001



## PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 3 Agustus 2018

Agung Pambudi

NIM: 145150200111062



## KATA PENGANTAR

Puji syukur kehadiran Allah SWT Yang Maha Esa penulis dapat menyelesaikan skripsi ini, jika bukan atas kehendaknya mustahil penulis dapat menyelesaikan skripsi untuk memperoleh gelar sarjana. Selain atas kehendak Allah SWT penulis menyadari banyak pihak yang membantu secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan seluruhnya, untuk itu penulis ucapkan terimakasih sedalam-dalamnya dan semoga Allah SWT membalas kebaikan dan bantuan yang telah diberikan pada penulis. Tak lupa Penulis ingin menyampaikan terimakasih khusus kepada:

1. Orang Tua penulis atas bimbingan, do'a dan kerja kerasnya untuk membiayai kuliah penulis sehingga penulis dapat menyelesaikan skripsi untuk mendapat gelar sarjana.
2. Keluarga penulis atas bantuan, do'a dan motifasinya.
3. Ibu Ari Kusyanti, S.T, MSc selaku pembimbing I atas bimbingan, saran dan motifasinya sehingga penulis dapat menyelesaikan skripsi ini.
4. Bapak Mahendra Data, S.Kom., M.Kom selaku pembimbing II atas bimbingan, saran dan motifasinya sehingga penulis dapat menyelesaikan skripsi ini.
5. Seluruh Dosen yang pernah membimbing dan memberi bekal ilmu pada penulis untuk menyelesaikan skripsi ini.
6. Teman-teman penulis atas bantuan, do'a dan motifasinya.

Penulis menyadari bahwa skripsi ini tidaklah sempurna untuk itu penulis membuka kesempatan bagi siapapun yang telah membaca skripsi ini untuk menyampaikan kritik dan saran demi kemajuan ilmu pengetahuan dan pengembangan penelitian selanjutnya.

Malang, 3 Agustus 2018

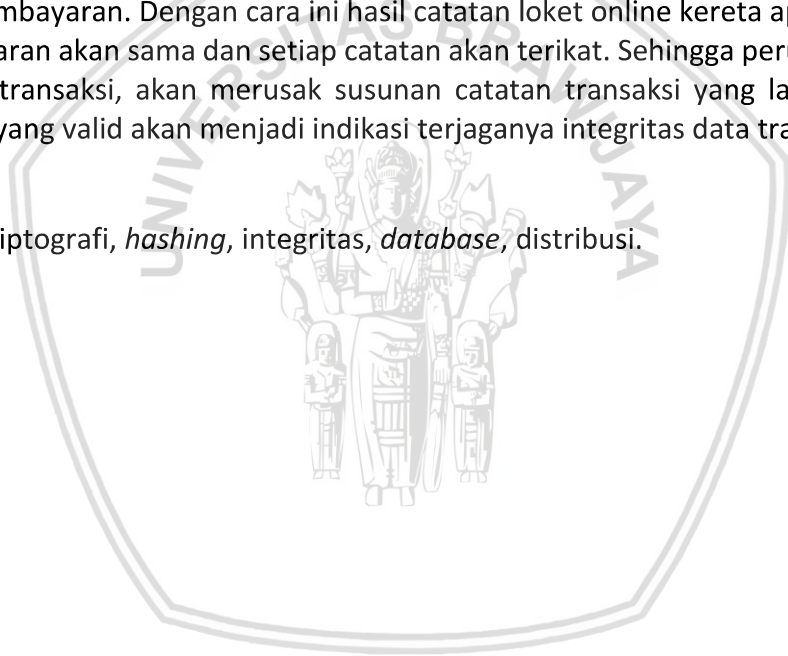
Penulis

agungpambudi@merahputih.id

## ABSTRAK

Dalam melakukan perdagangan, toko *online* membutuhkan gerai pembayaran sebagai perantara, agar uang yang pembeli miliki dapat diteruskan ke toko *online*. Saling terhubungnya transaksi antara loket *online* kereta api dengan beberapa gerai pembayaran, menuntut adanya distribusi *database*, hal ini akan menimbulkan masalah baru, karena memiliki celah keamanan. Yaitu manipulasi data transaksi. Sehingga diperlukan sebuah sistem yang dapat menjamin data transaksi tidak dapat dimanipulasi. Menggunakan metode hashing, sistem akan menghitung nilai digest dari transaksi yang masuk dari gerai pembayaran, untuk dikirimkan secara bergiliran dengan data transaksi ke server loket online kereta api. Kemudian akan dihitung nilai digest dari data transaksi. Serta digest catatan transaksi sebelumnya, untuk dicatatkan pada database server loket online, agar gerai pembayaran melakukan pencatatan yang sama pada database gerai pembayaran. Dengan cara ini hasil catatan loket online kereta api, dan gerai pembayaran akan sama dan setiap catatan akan terikat. Sehingga perubahan satu catatan transaksi, akan merusak susunan catatan transaksi yang lain. Susunan catatan yang valid akan menjadi indikasi terjaganya integritas data transaksi. Kata

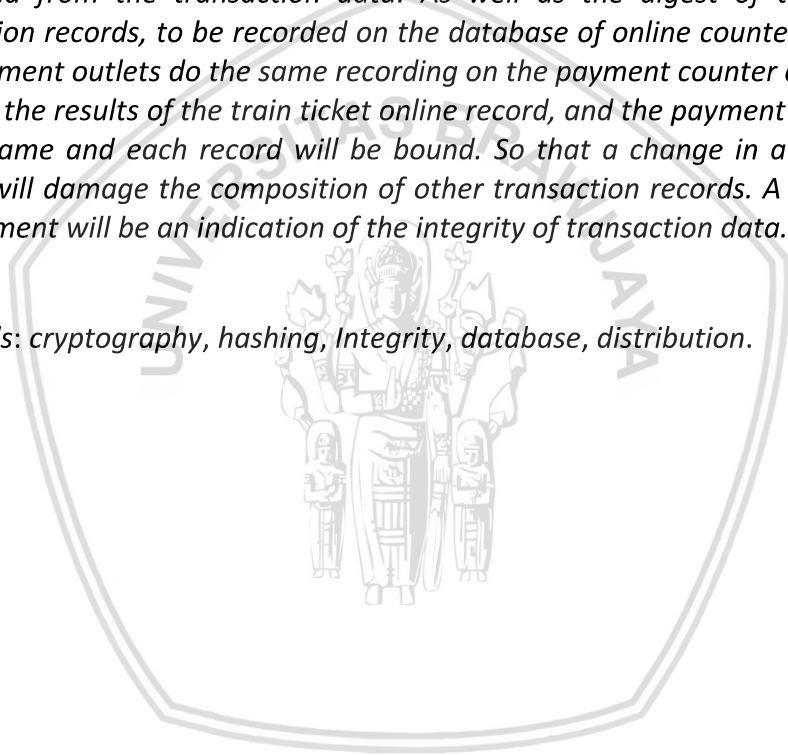
kunci: kriptografi, *hashing*, integritas, *database*, distribusi.



## ABSTRACT

*In trading, online stores need payment outlets as intermediaries, so that the money that the buyer has can be forwarded to an online store. The interconnection of transactions between online train counters and several payment outlets, demanding a database distribution will cause new problems, because it has a security gap, the gap is transaction data manipulation. For this reason the author makes a system that can guarantee transaction data cannot be manipulated. Using the hashing method, the system will calculate the digest value of the incoming transactions from the payment counter, to be sent in turn with transaction data to the online train ticket server. Then the digest value will be calculated from the transaction data. As well as the digest of the previous transaction records, to be recorded on the database of online counter servers, so that payment outlets do the same recording on the payment counter database. In this way the results of the train ticket online record, and the payment counter will be the same and each record will be bound. So that a change in a transaction record, will damage the composition of other transaction records. A valid record arrangement will be an indication of the integrity of transaction data.*

*Keywords: cryptography, hashing, Integrity, database, distribution.*



## DAFTAR ISI

PENGESAHAN .....	ii
PERNYATAAN ORISINALITAS .....	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	ix
DAFTAR GAMBAR .....	x
DAFTAR LAMPIRAN .....	xii
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan masalah .....	2
1.3 Tujuan .....	2
1.4 Manfaat.....	2
1.5 Batasan masalah .....	2
1.6 Sistematika pembahasan.....	2
BAB 2 LANDASAN KEPUSTAKAAN .....	4
2.1 Kajian Pustaka .....	4
2.2 Dasar Teori .....	5
2.2.1 Kriptografi .....	5
2.2.2 <i>Secure Hash Algorithm</i> .....	6
BAB 3 METODOLOGI .....	10
3.1 Studi Kepustakaan .....	10
3.2 Metode Penelitian .....	11
3.3 Perancangan .....	11
3.4 Implementasi .....	11
3.5 Pengujian Dan Pembahasan .....	11
3.6 Kesimpulan.....	12
BAB 4 Perancangan .....	13
4.1 Perancangan Algoritme .....	13

4.1.1 Cara Kerja Algoritme .....	13
4.1.2 Diagram Alir Algoritme.....	18
4.2 Perancangan Sistem.....	19
4.2.1 Cara Kerja Sistem .....	19
4.2.2 Sistem <i>Server Kereta Api</i> .....	21
4.2.3 Sistem <i>Server Gerai</i> .....	23
4.3 Perancangan Pengujian .....	24
BAB 5 IMPLEMENTASI .....	25
5.1 <i>Source Code Secure Hash Algorithm 1 (SHA-1)</i> .....	25
5.2 <i>Source Code Server Kereta</i> .....	27
5.3 <i>Source Code Server Gerai</i> .....	30
BAB 6 PENGUJIAN DAN PEMBAHASAN.....	33
6.1 Pengujian <i>Test Vector</i> .....	33
6.1.1 <i>Test Vector</i> Dengan Panjang Message Satu Blok .....	35
6.1.2 <i>Test Vector</i> Dengan Panjang Message Dua Blok.....	37
6.2 Pengujian Fungsionalitas .....	42
6.2.1 Transaksi dari gerai 1 .....	42
6.2.2 Transaksi dari gerai 2 .....	43
6.2.3 Transaksi dari gerai 3 .....	43
6.2.4 Transaksi dari gerai 4 .....	44
6.2.5 Hasil Pencatatan Transaksi <i>Server Kereta</i> .....	45
6.3 Pengujian Validitas Data Transaksi .....	45
6.4 Pengujian <i>Quality of Services</i> .....	48
6.4.1 Waktu Eksekusi Transaksi .....	48
6.4.2 <i>Throughput</i> .....	49
6.4.3 <i>Packet Loss</i> .....	49
BAB 7 KESIMPULAN DAN SARAN .....	50
7.1 Kesimpulan.....	50
7.2 Saran .....	50
DAFTAR PUSTAKA.....	51
LAMPIRAN .....	52

## DAFTAR TABEL

Tabel 2.1 Kajian Pustaka .....	5
Tabel 6.1 Hasil Pengujian Satu Blok .....	37
Tabel 6.2 Hasil Pengujian Dua Blok .....	42



## DAFTAR GAMBAR

Gambar 3.1 Langkah Penelitian .....	10
Gambar 4.1 Diagram Alir Modul <i>SHA-1</i> .....	18
Gambar 4.2 Gambaran Sistem .....	20
Gambar 4.3 <i>Database</i> Sistem.....	20
Gambar 4.4 Cara Kerja Sistem .....	21
Gambar 4.5 Diagram Alir Sistem <i>Server</i> Kereta .....	22
Gambar 4.6 Diagram Alir Sistem <i>Server</i> Gerai .....	23
Gambar 4.7 Perancangan Pengujian.....	24
Gambar 6.1 Hasil Perhitungan Satu Blok Bagian 1 .....	36
Gambar 6.2 Hasil Perhitungan Satu Blok Bagian 2 .....	38
Gambar 6.3 Hasil Perhitungan Dua Blok Bagian 1 .....	39
Gambar 6.4 Hasil Perhitungan Dua Blok Bagian 2 .....	39
Gambar 6.5 Hasil Perhitungan Dua Blok Bagian 3 .....	40
Gambar 6.6 Hasil Perhitungan Dua Blok Bagian 4 .....	41
Gambar 6.7 Transaksi Dari Gerai 1 .....	42
Gambar 6.8 <i>Database</i> Gerai 1 .....	43
Gambar 6.9 Transaksi Dari Gerai 2 .....	43
Gambar 6.10 <i>Database</i> Gerai 2.....	43
Gambar 6.11 Transaksi Dari Gerai 3 .....	44
Gambar 6.12 <i>Database</i> Gerai 3 .....	44
Gambar 6.13 Transaksi Dari Gerai 4 .....	44
Gambar 6.14 <i>Database</i> Gerai 4 .....	44
Gambar 6.15 <i>Database</i> <i>Server</i> Kereta .....	45
Gambar 6.16 Pengujian Validitas Data Transaksi 1 .....	47
Gambar 6.17 Pengujian Validitas Data Transaksi 2 .....	47
Gambar 6.18 Pengujian Validitas Data Transaksi .....	48
Gambar 6.19 Waktu Eksekusi Transaksi .....	48
Gambar 6.20 <i>Throughput</i> .....	49



Gambar 6.21 Transmisi <i>Packet</i> .....	49
---	----



## DAFTAR LAMPIRAN

Lampiran A. <i>Test Vector</i> Satu Blok .....	52
Lampiran B. <i>Test Vector</i> Dua Blok .....	56



## BAB 1 PENDAHULUAN

### 1.1 Latar Belakang

Transaksi jual beli telah dilakukan manusia sejak lama, jual beli merupakan cara untuk memenuhi kebutuhan hidup manusia. Seiring berkembangnya teknologi informasi, banyak perusahaan yang memanfaatkan teknologi untuk menjual produk atau jasa, salah satunya dengan mendirikan toko *online*. Toko *online* menyediakan berbagai macam barang dan jasa yang memudahkan konsumen untuk memilih barang atau jasa dan juga untuk melakukan pembayarannya, banyak pilihan metode pembayaran dalam berbelanja *online* tergantung pada kebijakan toko *online* masing-masing. Sebagian besar pembayaran *online* tidak ditangani langsung oleh toko *online*, namun dipercayakan pada beberapa gerai pembayaran yang telah menjadi mitra toko *online* tersebut. Hal ini menimbulkan masalah baru karena data transaksi tidak dibuat oleh toko *online*. Data transaksi dibuat oleh gerai pembayaran sehingga toko *online* hanya mendapat salinan data transaksi dari gerai mitranya, maka dari itu diperlukan sistem pengamanan data transaksi. Agar data yang telah dibuat oleh gerai pembayaran terjaga integritasnya dan tidak merugikan toko *online*. Mengingat fungsi data transaksi adalah sebagai bukti yang sah, sehingga data transaksi harus dapat dipertanggungjawabkan.

Fungsi *hash* memiliki peran mendasar dalam kriptografi modern, fungsi *hash* digunakan untuk integritas data bersama dengan skema tanda tangan digital, karena beberapa tujuan sebuah pesan biasanya di-*hash* terlebih dahulu, ide dasar dari fungsi *hash* kriptografi adalah nilai *hash* berfungsi sebagai gambaran representatif yang ringkas (terkadang disebut jejak, sidik jari digital, atau pesan yang olah) dari masukan *string*, sehingga kombinasi karakter unik dapat diidentifikasi sebagai *string* tersebut (Menezes, dkk., 1996).

Pada penelitian sebelumnya yang dilakukan oleh Satoshi Nakamoto, berjudul "*Bitcoin: A Peer-to-Peer Electronic Cash System*", mengulas bagaimana seorang Satoshi Nakamoto mengamankan sebuah data transaksi perdagangan dengan mata uang *Bitcoin*, menggunakan skema yang diberi nama *timestamp*. Maksud dari *timestamp* ialah, setiap stempel waktu menyertakan stempel waktu sebelumnya dalam *hash* yang baru, membentuk rantai, dengan masing-masing stempel waktu memperkuat yang sebelumnya (Nakamoto, 2008). Hal ini yang menjadi inspirasi penulis, untuk menyelesaikan masalah yang sebelumnya telah disampaikan. Menggunakan skema yang sama, data yang masuk di *server* gerai dilakukan penghitungan *hash*, dengan disertakan *hash* catatan sebelumnya. Sehingga catatan pada *database* membentuk sebuah rantai, satu catatan akan memperkuat cacatan lainnya. Sehingga catatan data transaksi yang dikirimkan ke *server* toko *online*, akan terjamin integritasnya. Begitu juga dengan *server* toko *online*, akan menerapkan skema yang sama agar integritas catatannya tetap terjaga. Jika Satoshi Nakamoto menerapkan skema tersebut pada *Bitcoin*, dalam penelitian ini penulis akan mengambil studi kasus toko *online* penjual tiket kereta api, agar lebih mudah dalam melakukan pemahaman.

## 1.2 Rumusan masalah

1. Algoritma *hashing* apa yang digunakan untuk mengamankan transaksi pada *database* terdistribusi ?
1. Bagaimana cara melakukan implementasi algoritme *hashing* ke dalam sistem pengamanan data transaksi ?
2. Bagaimana cara mengamankan data transaksi menggunakan metode *hashing* agar tidak dapat di manipulasi?

## 1.3 Tujuan

1. Memastikan algoritme *hashing* yang akan digunakan untuk mengamankan transaksi pada *database* terdistribusi telah sesuai.
2. Melakukan implementasi algoritme *hashing* ke dalam sistem pengamanan data transaksi.
3. Mengamankan data transaksi menggunakan metode *hashing* agar tidak dapat dimanipulasi.

## 1.4 Manfaat

1. Integritas data transaksi dapat dijaga dengan algoritme yang sesuai.
2. Sistem telah terintegrasi dengan algoritme *hashing* dan dapat berjalan sesuai fungsinya.
3. Data transaksi tidak dapat dimanipulasi oleh gerai-gerai resmi dan gerai-gerai mitra maupun pihak lain.

## 1.5 Batasan masalah

1. Karena sifat sistem perusahaan yang begitu kompleks, dan penulis hanya berfokus pada pengamanan data transaksi. penulis hanya merancang dan melakukan simulasi pertukaran data transaksi tanpa membuat sistem perusahaan secara penuh.
2. Sistem transaksi membutuhkan data hasil pemesanan, karena tidak adanya sistem pemesanan penulis akan mengisikan data pemesanan secara manual kedalam *database* pemesanan.
3. Penulis hanya memberikan gambaran jika sistem ini diterapkan pada kasus penjualan tiket kereta api namun tidak untuk menggantikan sistem yang saat ini digunakan sehingga penulis tidak melakukan penelitian bagaimana sistem penjualan tiket PT. Kereta Api sebenarnya bekerja.

## 1.6 Sistematika pembahasan

Untuk lebih mudah memahami isi dari skripsi ini, maka isi yang tertera pada skripsi ini dikelompokkan kedalam beberapa bab dengan sistematika penulisan sebagai berikut:

## **Bab I: Pendahuluan**

Berisi tentang latar belakang diangkatnya masalah data transaksi ini kedalam sebuah karya tulis ilmiah, perumusan masalah, tujuan dan manfaat dari penelitian ini, ruang lingkup penelitian, dan sistematika penulisan.

## **Bab II: Landasan Teori**

Bab ini berisikan teori yang berupa pengertian dan definisi yang diambil dari publikasi mengenai *SHA-1* beserta penjelasannya dan teori tentang kriptografi guna menunjang penelitian ini.

## **Bab III: Metodologi**

Bab ini berisikan tentang langkah-langkah yang akan dilakukan selama penelitian untuk dapat menyelesaikan masalah yang diangkat, dalam hal ini untuk merancang sistem pengamanan data transaksi yang diharapkan dapat dilakukan implementasi.

## **Bab IV: Perancangan**

Bab ini berisi penjelasan bagaimana penulis melakukan perancangan sistem pengamanan data transaksi dan penjelasan bagaimana data transaksi akan diamankan.

## **Bab V: Implementasi**

Bab ini berisi bagaimana rancangan yang telah penulis buat sebelumnya akan dilakukan implementasi kedalam kode program hingga sistem dapat dijalankan sebagaimana mestinya.

## **Bab VI: Pengujian**

Bab ini berisikan bagaimana sistem pengamanan data transaksi pada database terdistribusi yang sebelumnya telah dirancang dan dilakukan implementasi, akan diuji apakah penerapan metode *hashing* pada sistem ini dapat mengatasi masalah yang telah disebutkan sebelumnya.

## **Bab VIII: Kesimpulan**

Bab ini berisi kesimpulan dan saran yang penulis peroleh setelah melakukan penelitian berkaitan dengan penerapan metode *hashing* pada sistem pengamanan data transaksi, sebagaimana yang telah diuraikan pada bab-bab sebelumnya.

## BAB 2 LANDASAN KEPUSTAKAAN

### 2.1 Kajian Pustaka

Kemajuan teknologi adalah buah dari penelitian orang-orang terdahulu, satu penelitian akan menunjang penelitian lain dikemudian hari. Begitu juga penulis dalam melakukan penelitian ini ditunjang penelitian yang telah dipublikasikan sebagai berikut.

No	Nama Peneliti, Tahun, dan Judul	Persamaan	Perbedaan	
			Penelitian sebelumnya	Penelitian penulis
1	Satoshi Nakamoto, (2008), " Bitcoin: A Peer-to-Peer Electronic Cash System"	Menggunakan algoritme <i>Secure Hash Algorithm</i>	Implementasi pada Bitcoin dengan menggunakan block	Implementasi pada sistem pengamanan data transaksi dengan menggunakan <i>database</i>
2	Angger Ardyanto Pamungkas, (2006), " IMPLEMENTASI ALGORITMA MD5, SHA1, DAN RC4 UNTUK SISTEM KRIPTOGRAFI PADA APLIKASI MOBILE INTERNET BERBASIS JAVA"	Menggunakan algoritme <i>Secure Hash Algorithm 1</i>	Implementasi pada aplikasi <i>mobile</i> internet berbasis java	Implementasi pada sistem pengamanan data transaksi
3	Revati Raman Dewanagn, , Deepali Thombre, Kaushlendra Sharma (2015), " Implementation of Secure Hash Algorithm Using JAVA Programming"	Menggunakan algoritme <i>Secure Hash Algorithm 1</i>	Implementasi dengan menggunakan bahasa pemrograman JAVA	Implementasi dengan menggunakan bahasa pemrograman Python

Tabel 2.1 Kajian Pustaka



## 2.2 Dasar Teori

Dalam melakukan penelitian ini penulis mencoba menyelesaikan masalah yang ada dengan menggunakan metode dan algoritme sebagai berikut.

### 2.2.1 Kriptografi

Kriptografi adalah sebuah bidang studi matematika yang berkaitan tentang teknik pengamanan informasi dengan menggunakan rumus matematika yang telah dirancang sedemikian rupa agar dapat menjamin keamanan informasi. Kriptografi sendiri memiliki arti tulisan rahasia, diambil dari bahasa Yunani yaitu *cryptós* (rahasia) dan *gráphein* (tulisan). Kriptografi memiliki sejarah yang panjang, sekitar 4000 tahun yang lalu orang-orang Mesir kuno telah memakai teknik kriptografi untuk mengamankan informasi, kriptografi juga populer pada perang dunia kedua dan memegang peran penting bagi hasil peperangan, selain untuk fungsi militer kriptografi juga digunakan untuk layanan diplomatik dan pemerintahan suatu negara hingga saat ini, tentunya dengan teknik pengamanan yang lebih dikembangkan seiring berjalannya waktu. Pada awalnya kriptografi bekerja dengan cara menyandikan pesan asli yang dapat dimengerti kedalam bentuk yang tidak lagi dapat dimengerti, hal ini untuk menunjang komunikasi antar suatu golongan agar tidak dimengerti oleh golongan lain terlebih oleh musuh, namun dewasa ini kriptografi tidak lagi digunakan untuk sekedar menjaga kerahasiaan (*Confidentiality*) namun juga untuk menjaga integritas data (*Integrity*), Autentik (*Authentication*) (Menezes, dkk., 1996).

#### 2.2.1.1 Kerahasiaan (*Confidentiality*)

Kerahasiaan adalah tujuan kriptografi yang pertama, fungsinya untuk melindungi isi dari sebuah informasi agar tidak diketahui oleh orang yang tidak memiliki hak atau kewenangan. Untuk menjaga kerahasiaan dengan kriptografi ialah menggunakan metode enkripsi, metode enkripsi bekerja dengan cara mengubah isi dari sebuah informasi menggunakan perhitungan matematika serta menyisipkan sebuah kunci agar tidak lagi bisa dibaca, jika ingin membuka informasi tersebut haruslah mengetahui kunci yang telah diberikan dan melakukan perhitungan matematika agar informasi tersebut dapat dibaca kembali atau disebut dengan dekripsi. Pesan asli sebelum diubah disebut sebagai *plaintext* sedangkan pesan yang telah diubah dan tidak dapat dibaca disebut sebagai *ciphertext*. Kekuatan sebuah enkripsi terletak pada algoritme, algoritme menentukan bagaimana enkripsi bekerja serta metode untuk mengubah sebuah *plaintext* menjadi sebuah *ciphertext*, semakin baik algoritme maka semakin terjamin informasi yang dilindungi serta semakin susah untuk diretas. Enkripsi menurut kuncinya dibagi dua, yang pertama ialah enkripsi kunci simetris dan kunci yang digunakan untuk enkripsi dan dekripsi sama kemudian jenis yang kedua ialah kunci asimetris dan kunci yang digunakan untuk enkripsi adalah kunci publik dan kunci yang digunakan untuk dekripsi adalah kunci privat. Enkripsi tidak hanya dibedakan menurut kuncinya melainkan enkripsi juga dibedakan menurut metode

perhitungannya, yang pertama ialah enkripsi blok *Cipher* dan bentuk biner dari sebuah informasi akan dikelompokkan menjadi beberapa blok untuk kemudian dilakukan perhitungan perblok dan yang kedua ialah Stream *Cipher* dan bentuk biner dari sebuah informasi akan dihitung satu persatu (Menezes, dkk., 1996).

#### 2.2.1.2 Integritas (*Integrity*)

Integritas data adalah tujuan kriptografi yang kedua, fungsinya untuk mencegah terjadinya perubahan data yang tidak sah. Untuk menjamin integritas data, seseorang harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak yang tidak berwenang. Manipulasi data meliputi hal-hal seperti penyisipan, penghapusan, dan substitusi. Metode kriptografi untuk mendeteksi manipulasi data ialah *hashing*. *Hashing* bekerja dengan cara mengubah sebuah pesan atau dalam kriptografi disebut *message* menggunakan perhitungan matematika menjadi beberapa karakter tidak bisa dibaca yang mewakili nilai dari sebuah pesan yang disebut sebagai *digest*, *digest* tidak bisa diubah kembali menjadi *message* dan nilai *digest* akan berbeda pada tiap-tiap kombinasi pesan namun panjang *digest* akan selalu sama, hal ini berguna untuk membandingkan nilai *digest* pesan asli dengan pesan yang ingin diuji keasliannya, jika nilai *digest* sama berarti pesan tersebut asli dan tidak dimanipulasi namun jika nilai *digest* berbeda berarti pesan tersebut telah dimanipulasi, entah itu diubah atau disisipi sesuatu (Menezes, dkk., 1996).

#### 2.2.1.3 Autentikasi (*Authentication*)

Autentikasi adalah hal yang terkait dengan identifikasi. Fungsi ini berlaku untuk entitas dan informasi itu sendiri. Dua pihak yang akan berkomunikasi harus saling mengidentifikasi. Informasi yang disampaikan melalui jaringan harus diperiksa autentiknya seperti asal, tanggal asal, isi data, waktu dikirim, dan sebagainya, tujuannya untuk memastikan bahwa seseorang atau sesuatu adalah autentik atau asli. Autentik dibagi menjadi dua: autentik terhadap seseorang dan autentik terhadap objek. Autentik terhadap objek dilakukan untuk memastikan objek tersebut autentik dan tidak diubah oleh orang lain sementara autentik terhadap seseorang adalah memastikan pengirim data adalah orang yang benar dan tidak mengatasnamakan orang lain, autentik ini dapat menggunakan metode Digital Signature. Digital Signature bekerja dengan paduan antara *hashing* dan enkripsi, pertama-tama pengirim data akan melakukan penandatanganan atau signing pada dokumen sebelum dokumen tersebut dikirim, setelah dokumen dikirim penerima harus melakukan verifikasi atau verify untuk memastikan dokumen tersebut autentik (Menezes, dkk., 1996).

### 2.2.2 Secure Hash Algorithm

*Secure Hash Algorithm (SHA)* adalah sebuah algoritme *hashing* yang dapat mengubah pesan (*Message*) menjadi karakter tidak dapat dibaca yang merepresentasikan inti sari dari sebuah pesan (*Digest*), algoritme ini dirancang oleh National Institute of Standard and Technology (NIST) dan mulai



diperkenalkan pada bulan agustus tahun 1991 di Amerika Serikat dan merupakan bagian dari *Digital Signature Standard (DSS)* yang dijadikan standar pemrosesan informasi pemerintah Amerika Serikat atau disebut *U.S. Federal Information Processing Standard (FIPS)*. Beberapa versi *Secure Hash Algorithm* tergabung dalam *secure hash Standard (SHS)* (Dang, 2015). Dikembangkan untuk menggantikan algoritme-algoritme *hashing* sebelumnya yang dinilai telah tidak aman lagi untuk digunakan di lingkungan pemerintahan Amerika Serikat. *Secure Hash Algorithm* terus digunakan sampai sekarang dengan beberapa kali pembaruan dan saat ini versi yang paling baru ialah algoritme Keccak yang memenangkan sayembara dan berhak menyandang nama *Secure Hash Algorithm 3 (SHA-3)* serta resmi menggantikan *Secure hash Algorithm 2 (SHA-2)* sebagai standar algoritme *hashing*.

#### 2.2.2.1 Secure Hash Algorithm 1 (SHA-1)

*SHA-1* adalah salah satu varian dari keluarga *Secure Hash Algorithm (SHA)* yang merupakan bagian dari *Secure Hash Standard (SHS)* dengan fungsi yang sama yaitu mengubah pesan atau *message (M)* menjadi sebuah *digest*, *M* memiliki panjang sebanyak  $l$  bits ( $0 \leq l < 2^{64}$ ), algoritme ini menggunakan 1) penjadwalan pesan dari delapan puluh potongan pesan masing-masing 32 bit 2) lima variabel masing-masing 32 bit, dan 3) lima potongan *hash* masing-masing 32 bit. Sehingga akan *SHA-1* akan menghasilkan 160 bit *digest*. Potongan pesan pada saat penjadwalan akan diberi label  $W_0, W_1$  sampai  $W_{79}$ . Lima variabel kerja masing-masing akan diberi label  $a, b, c, d$ , dan  $e$ . potongan nilai *hash* akan diberi label  $H(i), H(i), \dots, H(i)$ , nilai  $H$  akan diproses sendiri-sendiri dan akan disatukan kembali menjadi sebuah *digest*. *SHA-1* hanya menggunakan satu penyimpanan sementara yaitu *TEMP (T)* (Dang, 2015).

#### 2.2.2.2 Message Padding SHA-1

*SHA-1* menggunakan blok dengan besaran 512 bit, tidak ada jumlah minimum atau maksimum yang membatasi jumlah pesan masukan, berapapun jumlah masukan akan tetap diproses dan menghasilkan *digest* dengan besaran 160 bit. Karena jumlah blok untuk pemrosesan adalah 512 bit dan masukannya belum tentu 512 bit, digunakan *message padding* yang akan menambah besaran bit pesan masukan menjadi kelipatan 512 agar dapat diproses oleh *SHA-1*. Cara kerja *padding* ialah menambahkan karakter "1" yang diikuti oleh karakter "0" sebanyak  $m$  buah serta 64 bit *integer* pesan asli sebelum dilakukan *padding* pada akhiran pesan untuk menghasilkan pesan dengan panjang 512 bit (Dang, 2015).

#### 2.2.2.3 Komputasi SHA-1

Berikut adalah fungsi untuk menghitung *SHA-1* dan setiap blok pesan,  $M(1), M(2), \dots, M(N)$ , diproses secara berurutan dengan langkah sebagai berikut:

Pertama-tama sebelum melakukan proses komputasi  $H$  diinisialisasi sebagai berikut:

$$H_0 = 67452301$$

$$H_1 = \text{EFCDAB89}$$

$$H_2 = 98BADCFE$$

$$H_3 = 10325476$$

$$H_4 = \text{C3D2E1F0}$$

Kemudian inisialisasi fungsi ( $f$ ) pada persamaan (2.1) dan Konstanta ( $k$ ) pada persamaan (2.2) sebagai berikut:

$$f_t(x,y,z)= \begin{cases} \text{Ch}(x, y, z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \leq t \leq 19 \\ \text{Parity}(x, y, z)=x \oplus y \oplus z & 20 \leq t \leq 39 \\ \text{Maj}(x, y, z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \leq t \leq 59 \\ \text{Parity}(x, y, z)=x \oplus y \oplus z & 60 \leq t \leq 79. \end{cases} \quad (2.1)$$

$$K_t = \begin{cases} 5a827999 & 0 \leq t \leq 19 \\ 6ed9eba1 & 20 \leq t \leq 39 \\ 8f1bbcdc & 40 \leq t \leq 59 \\ ca62c1d6 & 60 \leq t \leq 79. \end{cases} \quad (2.2)$$

Berikutnya lakukan komputasi:

For  $i=1$  to  $N$ :

{

1. Mempersiapkan penjadwalan pesan sesuai persamaan (2.3)  $\{W_t\}$ :

$$W_t = \begin{cases} M_t^{(i)} & 0 \leq t \leq 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \leq t \leq 79 \end{cases} \quad (2.3)$$

2. Inisialisasi lima variabel  $a, b, c, d$ , dan  $e$ , dengan nilai *hash* ( $i-1$ ):

$$a = H_0^{(i-1)}$$

$$b = H_1^{(i-1)}$$

$$c = H_2^{(i-1)}$$

$$d = H_3^{(i-1)}$$

$$e = H_4^{(i-1)}$$

3. For  $t=0$  to 79:

$$T = \text{ROTL}^5(a) + f_t(b, c, d) + e + K_t + W_t$$

$$e = d$$

$$d = c$$

$$c = \text{ROTL}^{30}(b)$$

$$b = a$$

$$a = T$$

4. Menghitung nilai  $H^{(i)}$ :

$$H_0^{(i)} = a + H_0^{(i-1)}$$

$$H_1^{(i)} = b + H_1^{(i-1)}$$

$$H_2^{(i)} = c + H_2^{(i-1)}$$

$$H_3^{(i)} = d + H_3^{(i-1)}$$

$$H_4^{(i)} = e + H_4^{(i-1)}$$

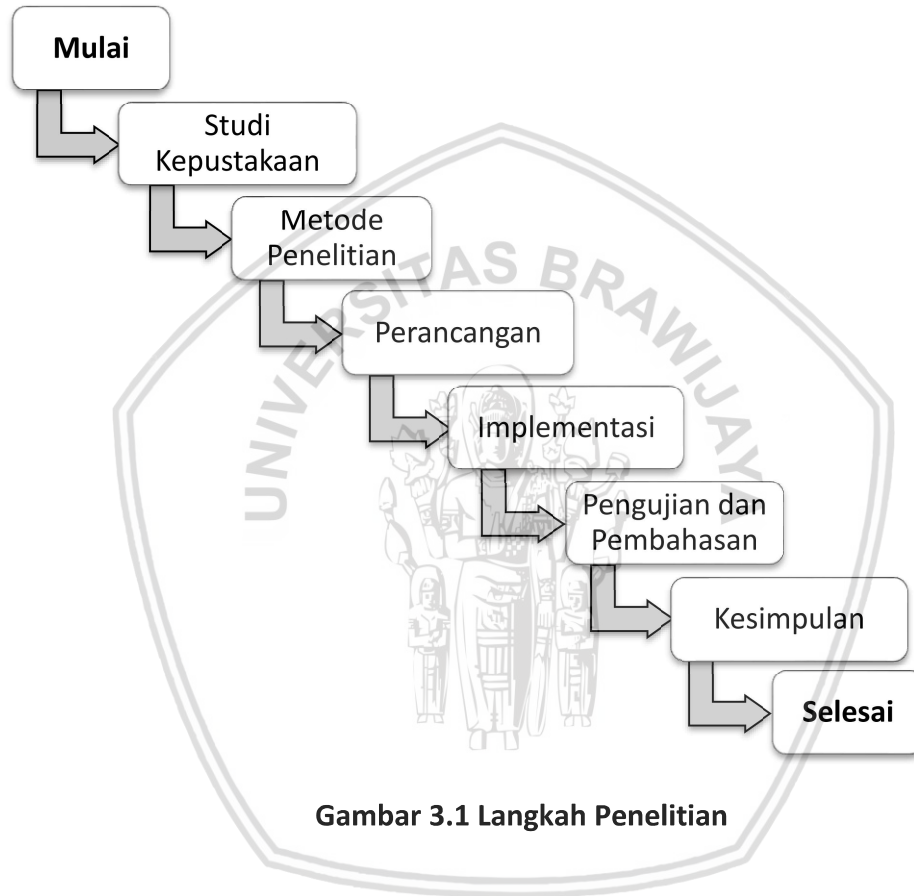
}

Setelah melakukan perhitungan empat langkah diatas akan menghasilkan 160-bit *digest* dengan cara menggabungkan nilai  $H_0$  sampai  $H_4$ .

$$H_0^{(N)} \parallel H_1^{(N)} \parallel H_2^{(N)} \parallel H_3^{(N)} \parallel H_4^{(N)}$$

## BAB 3 METODOLOGI

Bab Metodologi ini akan menjelaskan langkah-langkah yang dilakukan penulis selama penelitian, langkah-langkah dilakukan berurutan antar tahapnya dengan judul penelitian “Perancangan Sistem Pengamanan Data Transaksi Pada *Database* Terdistribusi Menggunakan Metode *Hashing*”. Berikut adalah tahapan yang dilakukan penulis dalam melakukan penelitian dan dapat dilihat pada Gambar 3.1.



Gambar 3.1 Langkah Penelitian

### 3.1 Studi Kepustakaan

Penelitian-penelitian yang telah dilakukan saat ini adalah buah dari penelitian yang telah lampau, melakukan penelitian saat ini akan menghasilkan penelitian baru dimasa depan baik oleh diri sendiri maupun orang lain, melakukan penelitian membutuhkan pemahaman yang mendalam terhadap penelitian yang relevan agar hasil penelitian dapat dipertanggung jawabkan kebenarannya, maka dari itu penulis melakukan studi kepustakaan terhadap penelitian yang telah ada yang relevan dengan penelitian penulis saat ini yaitu tentang Kriptografi, tujuan Kriptografi dan *Secure Hash Standard* yang penulis peroleh dari Buku dan publikasi ilmiah.

### 3.2 Metode Penelitian

Pada metode penelitian ini penulis menjelaskan tahapan-tahapan yang akan penulis lakukan selama penulis melakukan penelitian, tahapan-tahapan ini akan dilakukan setiap langkah secara berurutan agar penelitian menjadi teratur dan sistematis serta dapat menyelesaikan tujuan penelitian secara baik dan benar.

### 3.3 Perancangan

Dalam penelitian ini perancangan dimaksudkan untuk mencari rekayasa sistem yang paling sesuai agar kedepannya sistem yang telah dirancang dapat menyelesaikan masalah yang ada, sistem yang dirancang ialah sistem pengamanan data transaksi pada *database* terdistribusi yang menggunakan algoritme *hashing* sebagai metode pengamanannya. Penelitian ini tergolong penelitian implementatif, tujuan dari perancangan ini ialah mendapat model sistem yang paling sesuai untuk di implementasikan. Penulis akan melakukan penjelasan tentang kebutuhan sistem, alur kerja sistem dan skenario sistem.

### 3.4 Implementasi

Implementasi dilakukan dengan menerapkan rancangan yang sebelumnya telah dibuat kedalam kode program sesuai dengan model di perancangan hingga sistem siap untuk dijalankan, ada dua tahap implementasi, tahap pertama yaitu melakukan implementasi algoritme *Secure Hash Algorithm 1 (SHA-1)* kedalam kode program menggunakan bahasa pemrograman Python kemudian tahap kedua melakukan implementasi kode program *hashing* program tadi untuk menghasilkan *digest* dari data transaksi kemudian di masukan kedalam sistem *database* terdistribusi yang juga menggunakan bahasa pemrograman Python.

### 3.5 Pengujian Dan Pembahasan

Setelah sistem dirancang, dilakukan implementasi dan siap dijalankan perlu dilakukan untuk melihat apakah sistem dapat berjalan dengan baik dan juga menjadi bahan pembahasan apakah sistem dapat menyelesaikan masalah atau tidak. Pengujian dilakukan dengan terlebih dahulu membuat skenario uji untuk melakukan percobaan terhadap beberapa parameter yaitu:

1. Parameter pertama ialah akurasi hasil perhitungan *digest*, percobaan yang dilakukan untuk menguji parameter ini ialah dengan membandingkan *digest* yang dihasilkan dengan *test vector Secure Hash Algorithm 1 (SHA-1)* atau hasil *digest* program *Secure Hash Algorithm 1 (SHA-1)* lain yang sudah teruji untuk melakukan validasi bahwa algoritme *Secure Hash Algorithm 1 (SHA-1)* yang diterapkan telah sesuai dan dapat dipertanggung jawabkan.

2. Parameter kedua ialah kegunaan atau usability, percobaan dilakukan untuk mengetahui apakah sistem dapat bekerja dengan baik dan benar serta mengatasi masalah yang ada tanpa kendala dengan cara melakukan simulasi mulai dari sistem berjalan, data transaksi diubah menjadi *digest*, data dikirim kegerai yang lain hingga data disimpan kedalam *database* masing-masing.

### 3.6 Kesimpulan

Setelah semua tahap dilakukan dan dinilai sistem telah berjalan dengan baik dan benar, ditariklah kesimpulan yang didapat dari tahap awal hingga pengujian sehingga penulis dapat mengambil hikmah dari penelitian ini serta menuliskan saran yang bisa dijadikan rujukan bagi pembaca yang berniat melanjutkan penelitian ini agar nilai keilmuan dan pembaruan dapat berkembang.





## BAB 4 PERANCANGAN

Dalam membangun suatu sistem komputer diperlukan sebuah rancangan sebagai acuan untuk melakukan implementasi sebuah kode program menjadi sistem utuh, dalam penelitian ini penulis melakukan sebuah perancangan untuk menghasilkan rancangan yang nantinya akan menjadi acuan implementasi kode program. Pada penelitian ini ada dua fokus perancangan yang dibuat oleh penulis yaitu merancang bagaimana sistem secara keseluruhan bekerja dan merancang bagaimana bagian-bagian sistem bekerja.

### 4.1 Perancangan Algoritme

Perancangan algoritme dimaksudkan untuk mengenal lebih jauh algoritme yang akan diterapkan dalam penelitian ini, dalam perancangan algoritme ini penulis akan melakukan perhitungan manual bagaimana sebuah pesan akan diubah menjadi *digest* sesuai aturan algoritme *Secure Hash Algorithm 1 (SHA-1)* yang nantinya hasil perhitungan manual ini dapat dipakai sebagai *test vector* untuk pengujian nantinya, selain perhitungan manual penulis juga akan membuat diagram alir algoritme untuk dilakukan implementasi kedalam bahasa pemrograman.

#### 4.1.1 Cara Kerja Algoritme

Untuk mengetahui cara kerja algoritme penulis akan melakukan perhitungan manual untuk mengubah pesan menjadi *digest* dengan menggunakan algoritme *Secure Hash Algorithm 1 (SHA-1)*, pesan yang akan dirubah menjadi *digest* ialah “agung pambudi” berikut adalah langkah-langkah perhitungan *digest*:

1. Rubah pesan “agung pambudi” kedalam bentuk biner.

```
01100001011001110111010101101110011001110010000001110000011
000010110110101100010011101010110010001101001
```

2. Tambahkan “1” diakhir bilangan biner.

```
01100001011001110111010101101110011001110010000001110000011
0000101101101011000100111010101100100011010011
```

3. Tambahkan “0” sampai panjang bilangan sama dengan  $448 \bmod 512$ .

```
01100001011001110111010101101110011001110010000001110000011
000010110110101100010011101010110010001101001100000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
```

5. Bagi bilangan menjadi W sebanyak 80 bagian berformat *integer* dengan ketentuan bagian 17 sampai 80 dihitung dengan rumus ROTL1 (  $Wt-3 \oplus Wt-8 \oplus Wt-14 \oplus Wt-16$  ),  $\oplus$  adalah XOR, ROTL ialah menukar n bilangan paling akhir ke awal kumpulan bilangan.

W21 = 981582212



W22 = 3045708849  
W23 = 3324510289  
W24 = 1963164632  
W25 = 1527228491  
W26 = 3067199783  
W27 = 1603195777  
W28 = 1881624775  
W29 = 413139271  
W30 = 3557691313  
W31 = 1544200453  
W32 = 3787900010  
W33 = 2609814508  
W34 = 689897872  
W35 = 3155851502  
W36 = 2053652069  
W37 = 2586673669  
W38 = 1362903405  
W39 = 1989832437  
W40 = 1886521633  
W41 = 2625061527  
W42 = 854064395  
W43 = 378357267  
W44 = 2221688584  
W45 = 3636575897  
W46 = 3841169738  
W47 = 1793336872  
W48 = 3235177604  
W49 = 3189797310  
W50 = 384420268  
W51 = 3772559609  
W52 = 573608828  
W53 = 1161770890



W54 = 1254278142

W55 = 1148400237

W56 = 2398899785

W57 = 4231097736

W58 = 3357705093

W59 = 1087900789

W60 = 2101100909

W61 = 2121292156

W62 = 1551590538

W63 = 3675670829

W64 = 1298297914

W65 = 4251816075

W66 = 1339760880

W67 = 1353973369

W68 = 3511091913

W69 = 1640954070

W70 = 2448174729

W71 = 1680222211

W72 = 3571745344

W73 = 2698100735

W74 = 1036489783

W75 = 1979601505

W76 = 2711826850

W77 = 4074699105

W78 = 3927365809

W79 = 3355279375

W80 = 1225154518

6. Inisiasi nilai a,b,c,d dan e dengan ketentuan *SHA-1*

a = 67452301

b = EFCDAB89

c = 98BADCFE

d = 10325476

$$e = \text{C3D2E1F0}$$

7. Masukkan nilai  $t$ ,  $a$ ,  $b$ ,  $c$ ,  $d$  dan  $e$  kedalam rumus berikut lalu ulangi perhitungan sebanyak 80 kali,  $f$  adalah fungsi *SHA-1*,  $K$  adalah konstanta *SHA-1* dan ROTL adalah menukar  $n$  bilangan paling akhir ke awal kumpulan bilangan, perhitungan dilakukan dengan format biner 32 bit.

$$T = \text{ROTL}^5(a) + f_t(b, c, d) + e + K_t + t_t$$

$$e = d$$

$$d = c$$

$$c = \text{ROTL}^{30}(b)$$

$$b = a$$

$$a = T$$

8. Hasil dari perhitungan langkah 7 dalam format *integer*.

$$a = 2291184888$$

$$b = 3428386901$$

$$c = 4143244229$$

$$d = 126187710$$

$$e = 4202835696$$

9. Kemudian hitung dalam format biner 32 bit nilai  $H$  dengan rumus sebagai berikut:

$$H_0^{(i)} = a + H_0^{(i-1)}$$

$$H_1^{(i)} = b + H_1^{(i-1)}$$

$$H_2^{(i)} = c + H_2^{(i-1)}$$

$$H_3^{(i)} = d + H_3^{(i-1)}$$

$$H_4^{(i)} = e + H_4^{(i-1)}$$

10. Hasil dari perhitungan langkah 9 dalam format *integer* lalu dirubah ke format. *hexadecimal*.

$$H_0^{(i)} = 4023769081 = \text{efd5d7f9}$$

$$H_1^{(i)} = 3156653022 = \text{bc26b3de}$$

$$H_2^{(i)} = 2410660035 = \text{8fafc0c3}$$

$$H_3^{(i)} = 397921588 = \text{17b7cd34}$$

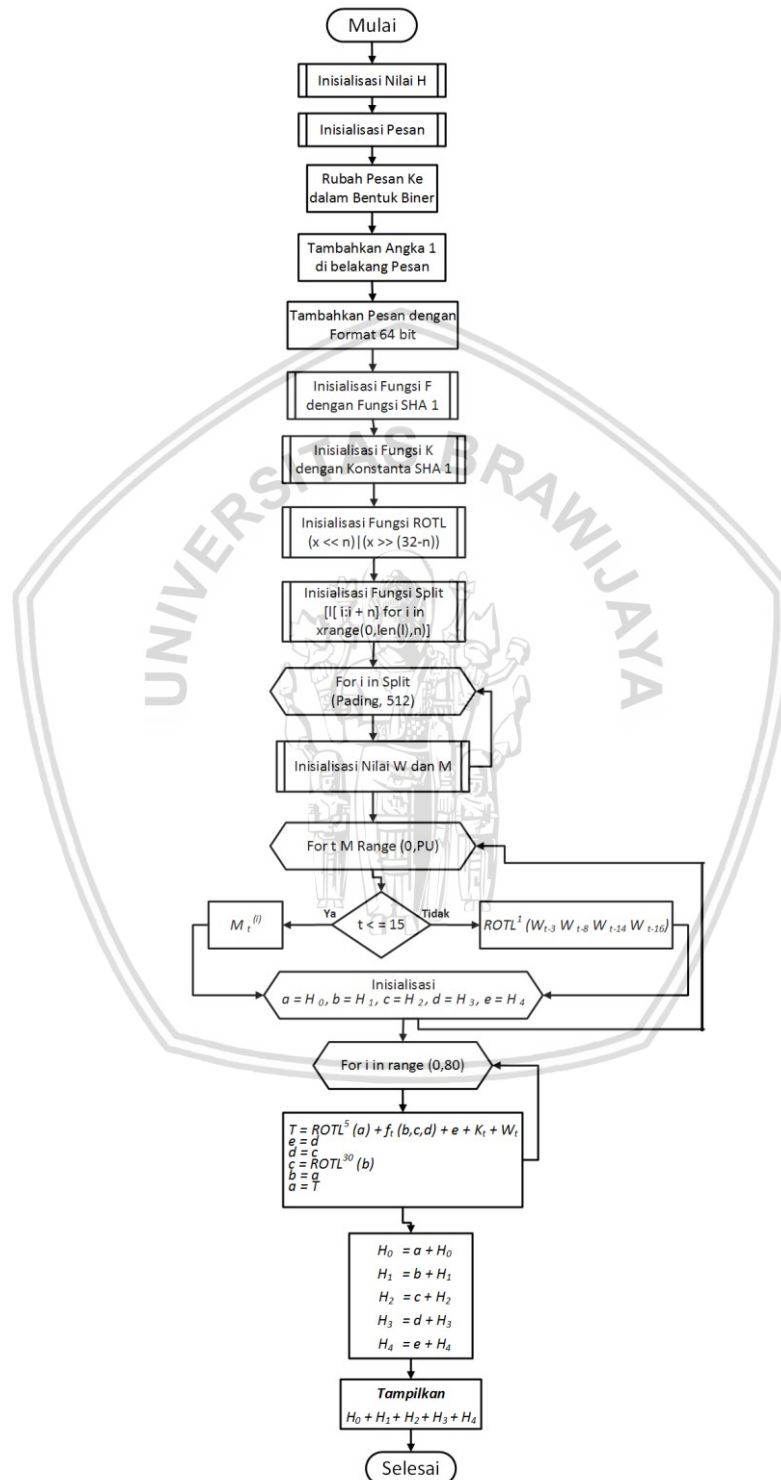
$$H_4^{(i)} = 3193245920 = \text{be5510e0}$$

11. *Digest* pesan “agung pambudi” ialah penggabungan dari nilai  $H$  dalam format *hexadecimal* sebagai berikut:

$$\text{efd5d7f9bc26b3de8fafc0c317b7cd34be5510e0}$$

#### 4.1.2 Diagram Alir Algoritme

Pada program ini penulis melakukan implementasi algoritme *SHA-1* dengan bahasa pemrograman Python, program ini digunakan sebagai modul penghitung *digest* yang akan digunakan *server* kereta dan *server* gerai.



Gambar 4.1 Diagram alir Modul *SHA-1*

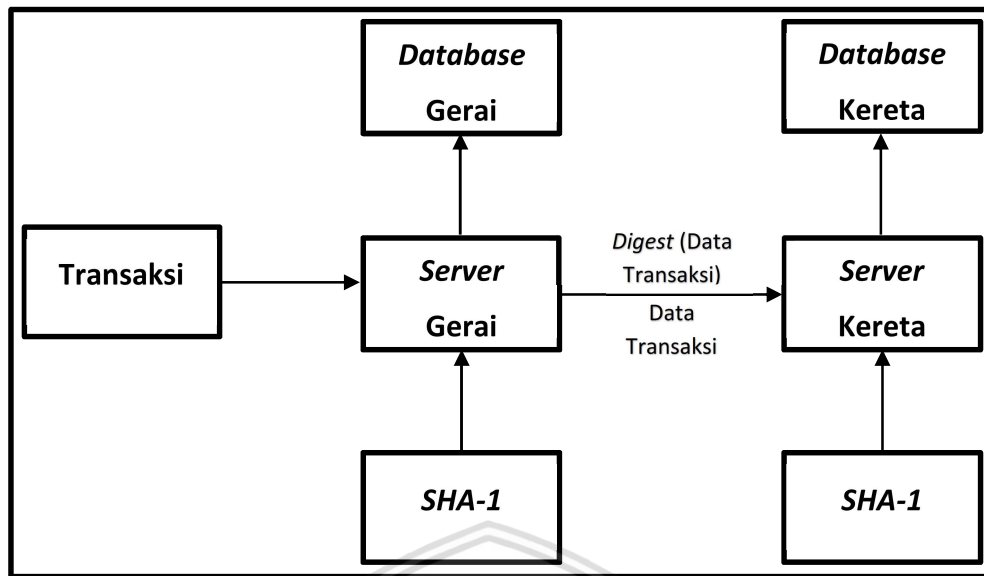
## 4.2 Perancangan Sistem

Perancangan sistem dimaksudkan untuk merancang alur kerja sistem untuk menentukan bagian-bagian yang nantinya akan membangun sistem dan akan dilakukan implementasi dengan kode program, penulis akan melakukan simulasi alur kerja sistem menggambar beberapa diagram yang dapat menjadi gambaran cara sistem bekerja, dengan dibuatnya perancangan ini diharapkan ketika melakukan implementasi tidak ada masalah yang timbul akibat ketidakjelasan alur kerja sistem.

### 4.2.1 Cara Kerja Sistem

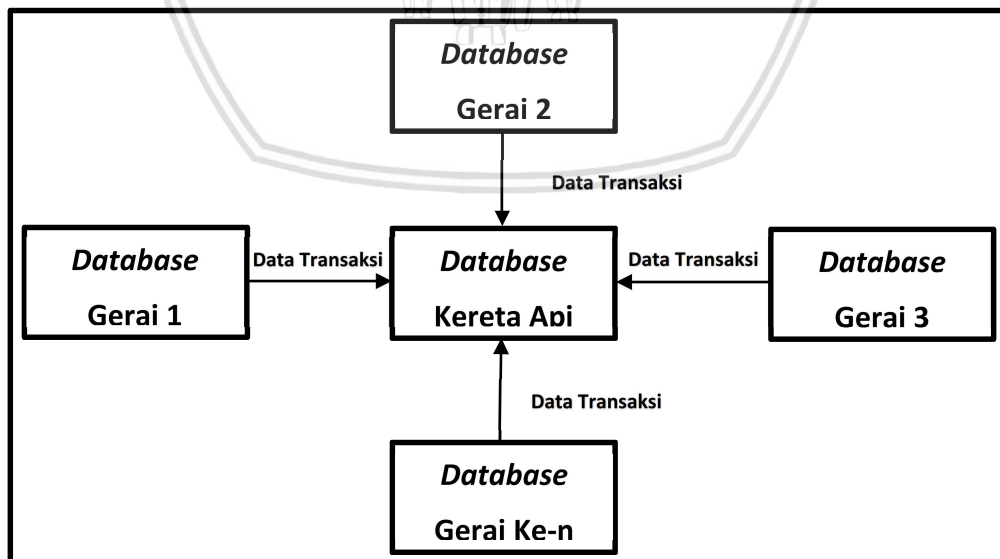
Tujuan dari Sistem Pengamanan Data Transaksi Pada *Database* Terdistribusi ialah untuk menjaga integritas data transaksi mengingat penjualan tiket kereta api saat ini didominasi dari gerai-gerai mitra secara *online* bukan lagi hanya di loket-loket setiap stasiun, gerai-gerai mitra yang jumlahnya tidak sedikit mutlak memerlukan distribusi *database* dari *database* kereta api dengan gerai-gerai lainnya yang menyebabkan duplikasi data transaksi. Hal ini dapat memunculkan masalah baru jika data transaksi tidak dijaga integritasnya mengingat data transaksi memiliki peran sebagai bukti transaksi penjualan tiket kereta, jika salah satu atau beberapa gerai diretas dan dirubah catatan data transaksinya sehingga data tersebut berbeda dengan data server kereta api atau sebaliknya, bukan hanya data transaksi di gerai-gerai mitra yang akan dipertanyakan integritasnya namun data transaksi di semua pihak akan diragukan integritasnya, maka dari itu diperlukan suatu sistem yang dapat menjaga integritas data baik di sisi *server* gerai, *server* kereta maupun pada saat pengiriman antar keduanya.

Berdasarkan fungsi yang telah dijelaskan sebelumnya, sistem membutuhkan 3 sub-sistem yang pertama sistem untuk menangani transaksi dan mengelola *database server* gerai, sistem untuk menangani data kiriman dari *server* gerai dan mengelola *server* kereta serta sistem untuk mengubah pesan menjadi *digest* untuk menunjang kinerja dua sistem yang lain seperti yang terlihat pada Gambar 4.1. Ketiga sub-sistem akan saling terhubung dengan skema sub-sistem *server* gerai akan melakukan transaksi kemudian data transaksi akan di dirubah menjadi *digest* dengan sub-sistem *hash* untuk ditambahkan pada data transaksi guna dicatatkan pada *database* serta dikirimkan ke *server* kereta sebelum pengiriman data transaksi yang dimaksudkan untuk membandingkan apakah *digest* yang telah dikirim nilainya sama dengan hasil perhitungan *digest* data transaksi yang dikirimkan setelahnya agar integritas data transaksi yang dikirimkan tetap terjaga, dan sistem *server* kereta akan mencatatkan data yang dikirim oleh *server* gerai pada *database* dengan ditambahkan *digest* dari data transaksi. Seluruh sub-sistem dibangun dengan bahasa pemrograman Python agar memudahkan pemrograman jaringan guna menunjang kinerja sistem.



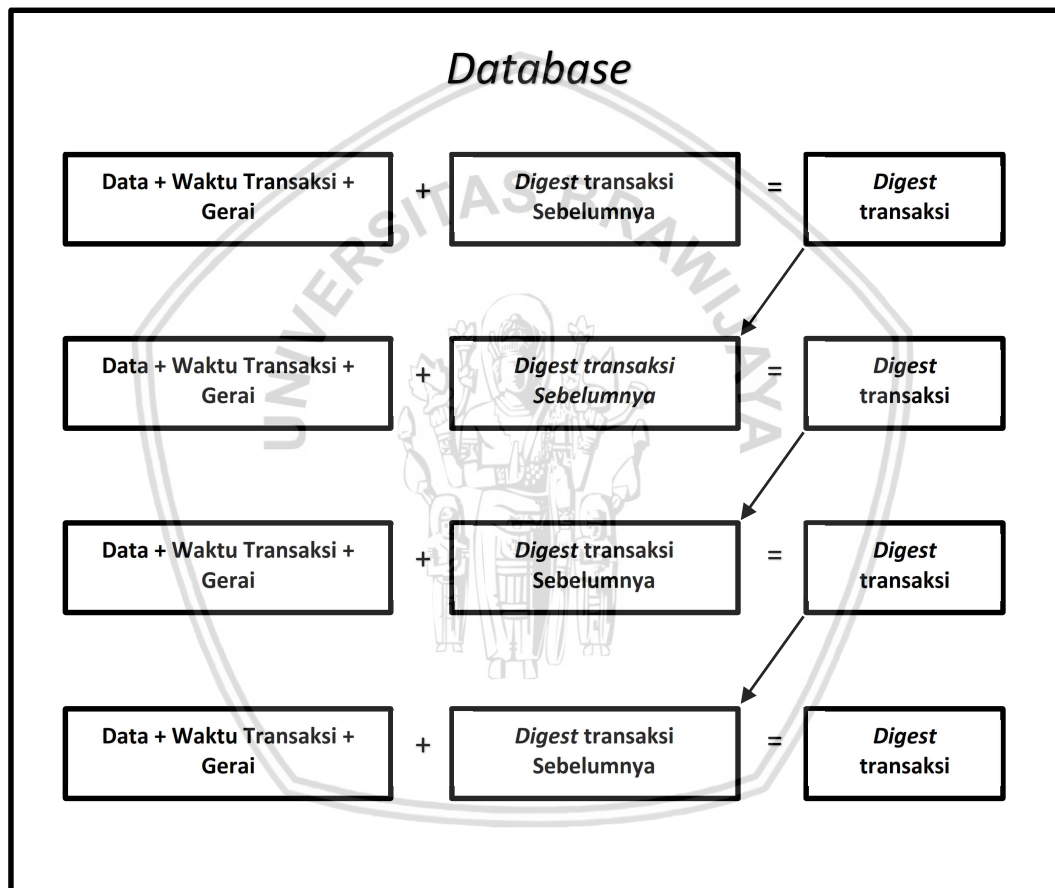
Gambar 4.2 Gambaran Sistem

Saat ini jumlah gerai mitra kereta api tidaklah sedikit sehingga nantinya sistem untuk gerai dibuat tidak terperinci dan bersifat umum agar dapat merepresentasikan banyak gerai penjual tiket kereta. Nantinya saat gerai-gerai melakukan transaksi, data akan disimpan kedalam *database* masing-masing dan dikirimkan ke *server* kereta api untuk disimpan dalam *database server* kereta api, itu artinya *database* kereta api akan mencatat data dari semua penjualan berbagai gerai tidak seperti *database* gerai yang hanya mencatat transaksi yang dilakukan gerai itu sendiri. Untuk menangani koneksi sistem gerai yang jumlahnya lebih dari satu maka sistem kereta akan bertindak sebagai *server* dan sistem gerai akan bertindak sebagai *client*, *server* akan menggunakan *protocol tcp* dan dibekali kemampuan multi-proses agar dapat menangani permintaan koneksi dari beberapa *client* dalam waktu bersamaan.



Gambar 4.3 Database Sistem

Cara kerjanya ialah data transaksi yang dihasilkan oleh proses transaksi sebelumnya akan ditambahkan cacatan waktu kapan transaksi itu berlangsung dan gerai mana yang melakukan transaksi serta nilai *digest* dari transaksi yang terakhir kali dilakukan, kemudian data tersebut akan dihitung nilai *digest*. Jika nilai *digest* telah ditemukan maka semua data akan dicatatkan pada baris baru *database* seperti yang terlihat pada gambar Gambar 4.2. catatan baru akan mengandung nilai *digest* catatan sebelumnya, jadi jika sedikit saja nilai data transaksi berubah maka nilai *digest* semua data-data transaksi setelahnya juga akan ikut berubah, hal ini bertujuan agar integritas data dapat terjaga dan jika ada manipulasi data dapat diketahui dengan mudah.



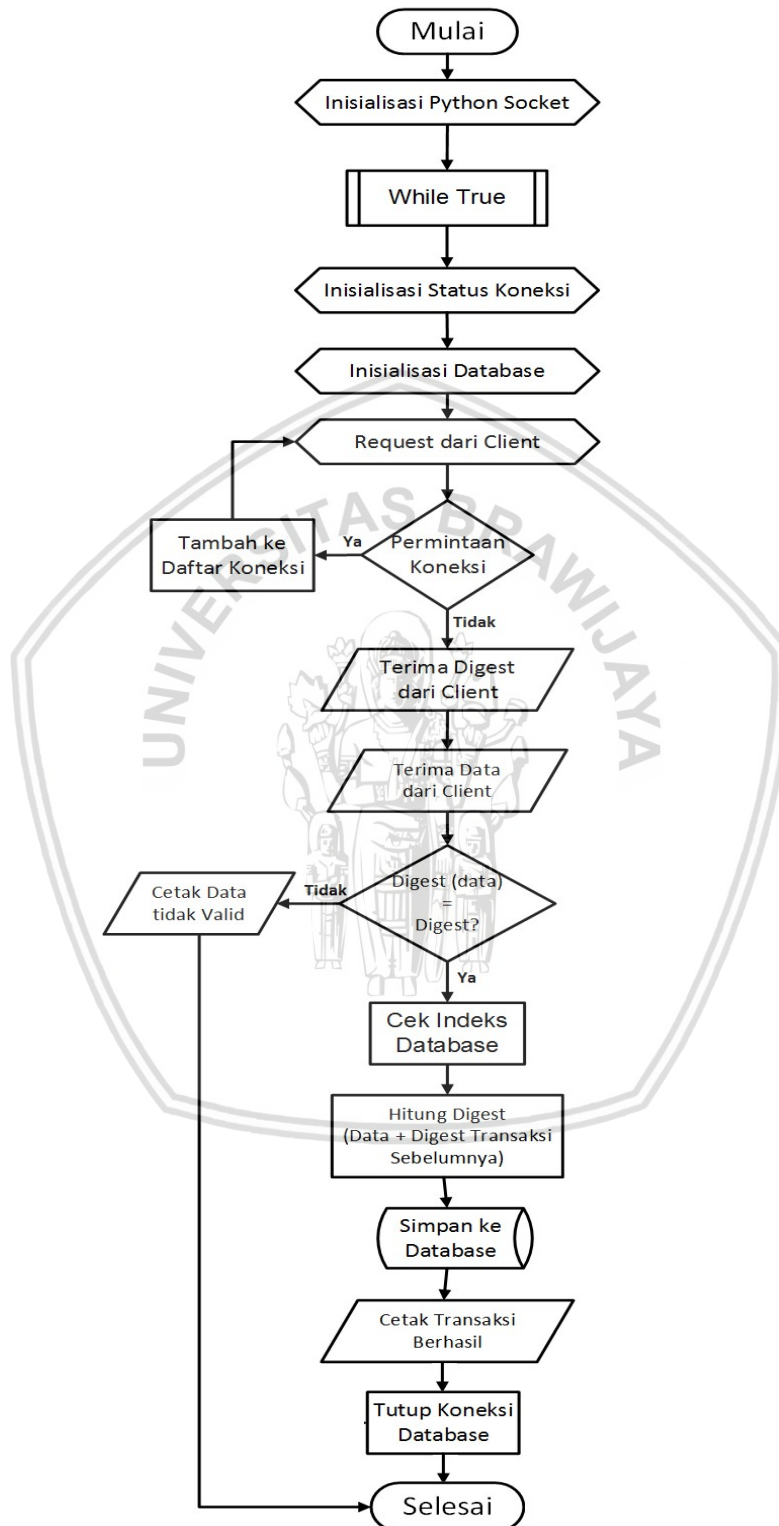
**Gambar 4.4 Cara Kerja Sistem**

#### 4.2.2 Sistem Server Kereta Api

Sistem dibangun dengan bahasa pemrograman Python dan beberapa *library* diperlukan di antaranya *socket* untuk pemrograman jaringan, *select* untuk multi proses dan PyMySQL untuk mengelola *database* MySQL. Sistem akan melakukan perulangan untuk menunggu jika ada *client* yang akan melakukan permintaan koneksi, jika permintaan koneksi bersifat *three way hand shaking* maka *server* akan memasukan *client* tersebut ke daftar, jika permintaan koneksi bersifat



permintaan layanan makan *server* akan menampung pengiriman data dan mengolah data, jika data berhasil diolah maka *server* akan mengirimkan pesan berhasil dan jika data gagal diolah maka *server* akan mengirimkan pesan gagal.

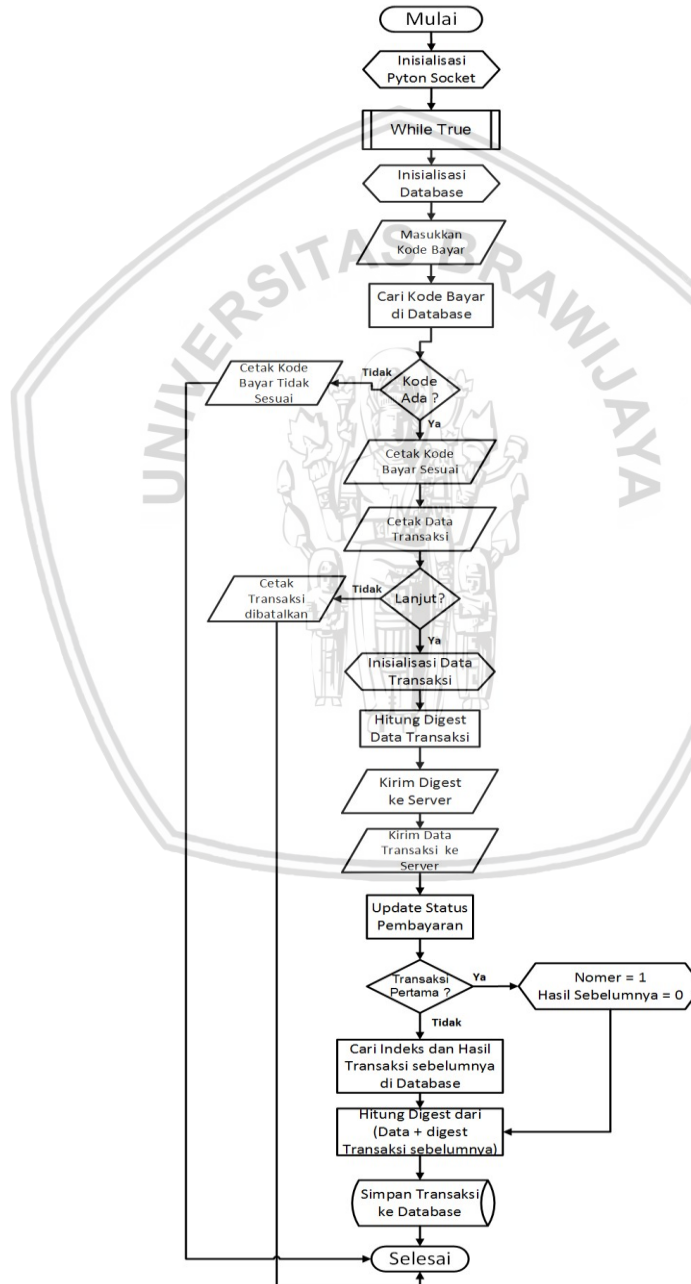


Gambar 4.5 Diagram alir Sistem *Server* Kereta



#### 4.2.3 Sistem Server Gerai

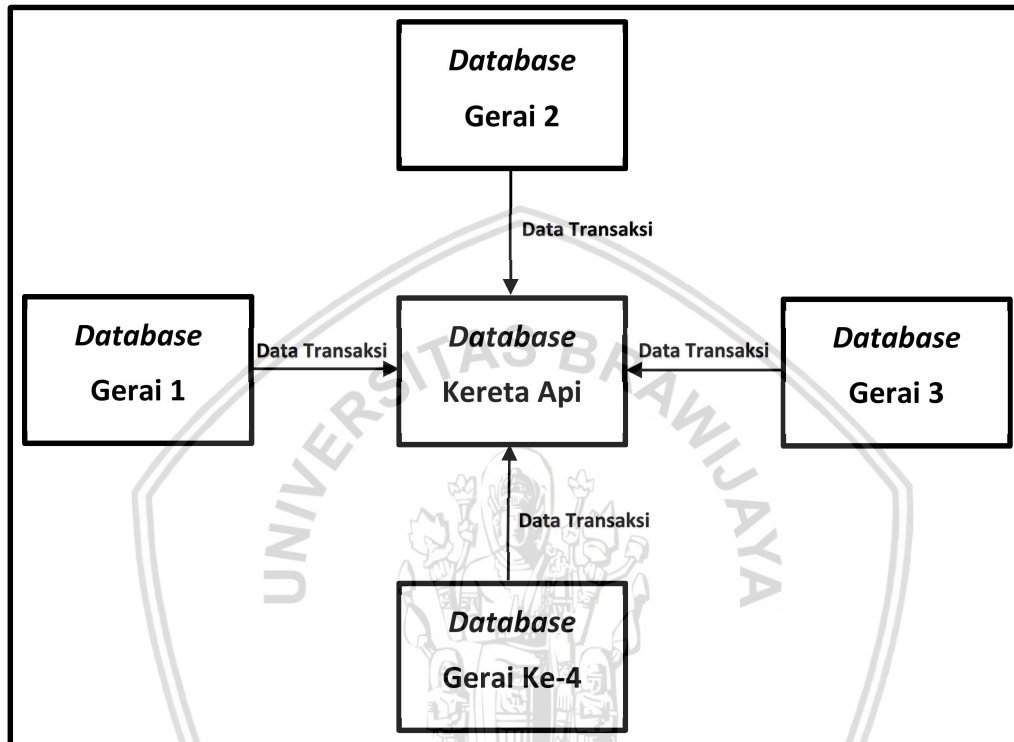
Sistem hampir sama dengan *Server Kereta* namun yang membedakannya ialah alur kerjanya dan tanpa *library select*, sistem akan melakukan perulangan untuk menunggu *client* yang akan melakukan transaksi, Sebelum melakukan transaksi dilakukan pengecekan terhadap kode bayar, jika kode bayar sesuai maka data transaksi akan dikirimkan ke *server* kereta dengan terlebih dahulu ditambahkan perhitungan *hash*, waktu transaksi dan nama gerai, jika transaksi dinyatakan berhasil oleh *server* kereta maka data transaksi beserta perhitungan *digest* akan dicatat dan sistem akan mencetak pesan transaksi berhasil.



Gambar 4.6 Diagram alir Sistem *Server Gerai*

### 4.3 Perancangan Pengujian

Setelah program dilakukan implementasi, penulis akan melakukan pengujian untuk memastikan sistem yang telah dibangun dapat berjalan dengan baik dan benar, ada empat parameter yang nantinya akan penulis ujikan terhadap sistem, yang pertama adalah ketepatan hasil perhitungan *digest* dari modul *SHA-1* dengan cara membandingkan hasil perhitungan program dengan *test vector* resmi.



Gambar 4.7 Perancangan Pengujian

Pengujian yang kedua ialah pengujian terhadap fungsionalitas sistem, karena sistem ini adalah sistem pengamanan data transaksi pada *database* terdistribusi maka penguji akan mencoba melakukan transaksi dari empat gerai seperti yang terlihat pada Gambar 4.6 yang nantinya keempat gerai tersebut akan mendistribusikan data transaksinya ke *server* kereta, penulis akan menilai apakah sistem dapat mengamankan data transaksi sebagaimana telah dijelaskan sebelumnya atau sistem gagal menjalankan fungsinya tersebut. pengujian ketiga adalah pengujian *quality of services*, pengujian ini akan mengukur kualitas dari layanan sistem, kemudian pengujian keempat dan yang terakhir adalah pengujian validitas data transaksi, karena tujuan dari sistem ini ialah menjaga integritas data transaksi maka validitas data transaksi perlu diuji untuk membuktikan bahwa sistem ini dapat mencapai tujuannya dan catatan data transaksi yang dihasilkan terbukti valid.

## BAB 5 IMPLEMENTASI

Setelah sebelumnya penulis membuat rancangan sebagai acuan untuk implementasi, pada bab ini penulis akan melakukan implementasi rancangan tersebut kedalam bahasa pemrograman Python. Sistem terdiri dari tiga sub-sistem yaitu sistem *Secure Hash Algorithm 1 (SHA-1)*, sistem *Server Kereta*, Sistem *Server Gerai* yang ketiganya jika disatukan akan membentuk Sistem Pengamanan Data Transaksi Pada *Database Terdistribusi*.

### 5.1 Source Code Secure Hash Algorithm 1 (SHA-1)

Kode program ini adalah implementasi dari algoritme *SHA-1* yang dipublikasikan oleh National Institute of Standards and Technology (*NIST*) yang ditulis dengan bahasa pemrograman Python. Berfungsi sebagai modul *hashing*

Pertama-tama deklarasikan variabel *H0* sampai *H4* dengan nilai yang telah ditentukan oleh pembuat algoritme, untuk merubah sebuah pesan menjadi *digest* dengan panjang yang selalu tetap. Pertama-tama pesan akan diambil dari parameter *message* pada fungsi *GenerateDigest* kemudian akan dirubah kedalam bentuk biner kemudian akan ditambahkan angka satu dibelangan kemudian pesan akan digenapkan menjadi kelipatan  $512 \bmod 448$  dengan menambahkan angka nol serta ditambahkan biner asli dengan format 64 bit sehingga pesan menjadi kelipatan 512, proses ini disebut dengan *padding*. Setelah pesan dilakukan *padding* kemudian pesan akan dibagi menjadi 80 bagian dengan bagian pertama sampai enam belas merupakan urutan pertama pesan dengan panjang 32 bit sedangkan urutan tujuh belas sampai delapan puluh adalah hasil perhitungan fungsi *ROTL* yang telah dideklarasikan sebelumnya dan telah ditentukan oleh pembuat algoritme. Setelah pesan dibagi menjadi 80 bagian. Kemudian dideklarasikan variabel *a*, *b*, *c*, dan *d* dengan nilai dari variabel *H1*, *H2*, *H3* dan *H4* untuk kemudian dihitung dengan rumus pada baris ke tujuh puluh tujuh sampai delapan puluh dua dengan menyertakan fungsi *fuction* dan fungsi *constant* yang telah di deklarasikan sebelumnya dan telah ditentukan oleh pembuat algoritme. Kemudian hasil dari perhitungan akan dimasukan kembali kedalam variabel *H0* sampai *H4* seperti pada baris delapan puluh empat sampai delapan puluh delapan. kemudian nilai dari variabel *H0* sampai *H4* akan digabungkan sehingga diperoleh nilai *digest*.

Algoritme 1 : Fungsi <i>SHA-1</i>	
1	def GenerateDigest(message):
2	
3	H0 = 0x67452301
4	H1 = 0xEFCDAB89
5	H2 = 0x98BADCFE
6	H3 = 0x10325476
7	H4 = 0xC3D2E1F0
8	
9	#Ubah Message kedalam bentuk biner
10	biner = ""
11	

```

12     biner = biner + (''.join("{0:08b}".format(ord(x), 'b') for x
13     in message))
14
15     #Tambah 1 dibelakang pesan yang telha diubah kedalam biner
16     padded = biner+"1"
17
18     #Tambah angka 0
19     While True
20         padded=padded+"0"
21         if (len(padded)%512 == 448%512):
22             break
23
24     #Tambah 64bit pesan asli
25     padded=padded+'{0:064b}'.format(len(biner))
26
27     #Function
28     def funct(x, y, z, i):
29         if 0 <= i <= 19:
30             f = (x & y) ^ ((~x) & z)
31         elif 20 <= i <= 39:
32             f = x ^ y ^ z
33         elif 40 <= i <= 59:
34             f = (x & y) ^ (x & z) ^ (y & z)
35         elif 60 <= i <= 79:
36             f = x ^ y ^ z
37         return f
38
39     #Constants
40     def K(i):
41         if 0 <= i <= 19:
42             k = 0x5A827999
43         elif 20 <= i <= 39:
44             k = 0x6ED9EBA1
45         elif 40 <= i <= 59:
46             k = 0x8F1BBCDC
47         elif 60 <= i <= 79:
48             k = 0xCA62C1D6
49         return k
50
51     #Sn(X)
52     def ROTL(X, n):
53         rotate = (( X << n) | (X >> (32 - n))) & 0xffffffff
54         return rotate
55
56     def split(l, n):
57         s = [l[i:i+n] for i in xrange(0, len(l), n)]
58         return s
59
60     for i in split(padded, 512):
61         w = [0]*80
62         M = split(i, 32)
63         for t in range(0, 80):
64             if t <= 15 :
65                 #Mt(i)
66                 w[t] = int(M[t], 2)
67             else :
68                 #ROTL1=( tt3^tt8^tt14^tt16 )

```

	$w[t] = \text{ROTL}((w[t-3] \wedge w[t-8] \wedge w[t-14] \wedge w[t-16])),$
68	1)
69	
70	#Initialize the five working variabels
71	a = H0
72	b = H1
73	c = H2
74	d = H3
75	e = H4
76	
77	for i in range(0, 80):
	T = ROTL(a,5) + funct(b,c,d,i) + e + K(i) + w[i] &
78	0xffffffff
79	e = d
80	d = c
81	c = ROTL(b,30)
82	b = a
83	a = T
84	
85	H0 = a + H0 & 0xffffffff
86	H1 = b + H1 & 0xffffffff
87	H2 = c + H2 & 0xffffffff
88	H3 = d + H3 & 0xffffffff
89	H4 = e + H4 & 0xffffffff
90	
91	Digest
92	'%08x'%H0+'%08x'%H1+'%08x'%H2+'%08x'%H3+'%08x'%H4
93	return Digest

## 5.2 Source Code Server Kereta

Program *Server Kereta* berfungsi untuk mengelola data transaksi serta mengelola *database*. Program ini nantinya dijalankan pada *server Kereta Api*, dalam penelitian ini penulis menggunakan MySQL sebagai *database* dan PyMySQL sebagai *library* pengelola *database* karena program ini menggunakan bahasa pemrograman Python serta *library select* untuk multi proses agar *server* dapat melayani beberapa *client* secara bersamaan, komunikasi antara *client* dan *server* menggunakan *protocol tcp* dan diasumsikan alamat IP dari *server* adalah 192.168.56.101 dan komunikasi menggunakan port 7500. Pada implementasi sesungguhnya bisa jadi *database* yang digunakan atau alamat *server* yang digunakan berbeda, dalam penelitian ini penulis hanya melakukan simulasi untuk keperluan ilmu pengetahuan.

Cara kerja program ini ialah akan memanfaatkan *library socket, select* Python sebagai penunjang komunikasi jaringan kemudian *library PyMySQL* untuk mengelola *database* MySQL dan juga memanfaatkan program *SHA-1* sebagai penghitung *digest* dari data transaksi, program ini akan menunggu permintaan koneksi dari *server* gerai yang akan melakukan transaksi, ketika ada *server* gerai yang meminta koneksi maka *server* kereta akan mengizinkan dan menambahkan *server* gerai tersebut kedalam daftar koneksi, *server* kereta akan melakukan koneksi dengan *database* untuk mempersiapkan transaksi dan koneksi dari *server*

gerai akan segera dilayani. *Server* kereta akan menunggu pengiriman data transaksi dari *server* gerai, pertama-tama *server* gerai akan mengirim hasil perhitungan *digest* dari data transaksi terlebih dahulu, setelah *digest* disimpan oleh *server* kereta kemudian data transaksi akan dikirim dan nantinya data transaksi akan dihitung ulang oleh *server* kereta serta dicocokkan hasil perhitungannya dengan *digest* yang telah disimpan sebelumnya, hal ini mencegah adanya manipulasi data saat data transaksi di transmisikan. Kemudian *server* kereta akan memeriksa indeks dari *database* untuk menentukan nomor catatan baru dan menentukan *hashing* catatan sebelumnya untuk menentukan *hashing* dari data transaksi yang sedang diproses, *server* kereta akan memanggil program *SHA-1* untuk menghitung *digest* dari data transaksi beserta *digest* data transaksi sebelumnya dan mencatatkannya kedalam *database*. Setelah transaksi selesai dicatatkan maka *server* kereta akan memberikan konfirmasi kepada *server* gerai bahwa transaksi telah berhasil dilakukan.

Algoritme 2 : Fungsi <i>Server</i> Kereta			
1	import socket		
2	import select		
3	import pymysql		
4	from sha1 import GenerateDigest		
5			
6	sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)		
7			
8	sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)		
9			
10	sock.bind(('', 7500) )		
11			
12	sock.listen(10)		
13			
14	list_koneksi = [sock]		
15			
16	def send_to_all_clients(conn, msg):		
17	for client in conn.clients :		
18	client.connection.send(msg)		
19			
20	while True :		
21	inputready, outputready, errorready	=	
	select.select(list_koneksi, [], [])		
22			
23			
24	for n in inputready :		
25	db	=	
	pymysql.connect("localhost","root","br89q","transaksi" )		
26	cursor = db.cursor()		
27			
28	if n == sock :		
29	conn, addr = sock.accept()		
30	list_koneksi.append(conn)		
31	else :		
32	try :		
33	data = n.recv(100)		
34	try :		
35	hash1 = data		
36	n.send("Sedang Diproses oleh Server.....")		



```

37         print "Transaksi Sedang Dilakukan"
38         data = n.recv(250)
39
40         nama,id,tipe,tempatduduk,nomortiket,kereta,kelas,
41         berangkat,tiba,statusbayar,harga,kodebayar,waktubayar,gerai      =
42         data.split("$?!")
43
44         print (gerai+" Memulai Transaksi")
45         hash2 =
46         GenerateDigest(nama+id+tipe+tempatduduk+
47         nomortiket+kereta+kelas+berangkat+tiba+statusbayar+harga+kodebay
48         ar+waktubayar+gerai)
49
50         if hash1==hash2:
51             hashsebelum = ""
52
53             sql      =      "SELECT      max(nomor)      FROM
54             pembayaran;"
55             cursor.execute(sql)
56             b=cursor.fetchall()
57             for row in b :
58                 nomor = (row[0])
59
60             if nomor != None:
61                 sql = "SELECT * FROM pembayaran WHERE
62                 nomor=%s;"
63                 cursor.execute(sql,(nomor))
64                 b=cursor.fetchall()
65                 for row in b :
66                     hashsebelum = (row[16])
67                     nomor= str(int(nomor)+1)
68
69             elif nomor == None:
70                 nomor="1"
71                 hashsebelum = "0"
72
73             hashsekarang=
74             GenerateDigest(nomor+nama+id+tipe+
75             tempatduduk+nomortiket+kereta+kelas+berangkat+tiba+statusbayar+h
76             arga+kodebayar+waktubayar+gerai+hashsebelum)
77
78             sql = "INSERT INTO pembayaran (nomor,
79             nama, id, tipe, tempatduduk, nomortiket, kereta, kelas, berangkat,
80             tiba, statusbayar, harga, kodebayar, waktubayar, gerai,
81             hashsebelum, hash) VALUES
82             (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
83
84             cursor.execute(sql,((nomor,nama,id,tipe,tempatduduk,
85             nomortiket,kereta,kelas,berangkat,tiba,statusbayar,harga,kodebay
86             ar,waktubayar,gerai,hashsebelum,hashsekarang)))
87             db.commit()
88             n.send("Transaksi Berhasil")
89             print ("Transaksi "+gerai+" Berhasil")
90
91             else :
92                 print "Integritas Data Tidak Dipercaya
93                 Transaksi Dibatalkan"
94
95             except ValueError:
96                 list_koneksi.remove(n)

```



75	n.close()
76	print ("Terjadi Kesalahan, Koneksi ke client "+gerai+" diputus")
77	except socket.error:
78	list_koneksi.remove(n)
79	n.close()
80	print ("Koneksi ke client "+gerai+" diputus")
81	db.close()

### 5.3 Source Code Server Gerai

Program *Server Gerai* berfungsi untuk melakukan transaksi dari data pemesanan serta mengelola *database*. Program ini nantinya dijalankan pada *server* gerai-gerai mitra penjualan, dalam penelitian ini penulis menggunakan MySQL sebagai *database* dan PyMySQL sebagai *library* pengelola *database* karena program ini menggunakan bahasa pemrograman Python, komunikasi antara *client* dan *server* menggunakan *protocol tcp* dan diasumsikan alamat IP dari *server* adalah 192.168.56.102 sampai 192.168.56.254 sesuai kebutuhan dan komunikasi menggunakan port 7500. Pada implementasi sesungguhnya bisa jadi *database* yang digunakan atau alamat *server* yang digunakan berbeda, dalam penelitian ini penulis hanya melakukan simulasi untuk keperluan ilmu pengetahuan.

Cara kerja program ini ialah akan memanfaatkan *library socket* Python sebagai penunjang komunikasi jaringan kemudian *library MySQL* untuk mengelola *database MySQL* dan juga memanfaatkan program *SHA-1* sebagai penghitung *digest* dari data transaksi, inisialisasi koneksi diatur agar program dapat berkomunikasi dengan *server* kereta dengan menuliskan alamat ip dari *server* kereta, saat program dijalankan maka koneksi ke *server* kereta akan dibuat kemudian program akan melakukan koneksi ke *database* setelah itu program akan meminta pengguna memasukkan kode bayar ketika akan melakukan transaksi, program akan mencari data pemesanan yang memiliki kode program sama dengan kode program yang dimasukan pengguna, jika kode program ada maka program akan mencetak data pemesanan untuk klarifikasi namun jika kode bayar tidak ditemukan maka program akan membatalkan transaksi. Setelah pengguna melakukan konfirmasi bahwa data transaksi benar dan bersedia melanjutkan transaksi maka *server* gerai akan menghitung *digest* dari data transaksi untuk dikirimkan ke *server* kereta kemudian data transaksi akan dikirimkan pada transmisi berikutnya. Program akan menerima status pembayaran dari *server* kereta dan *server* gerai akan mencetak status tersebut. Jika pembayaran dinyatakan berhasil oleh *server* kereta dan data telah dicatat pada *database* maka data transaksi akan dihitung sebelum dicatatkan pada *database server* gerai dengan nomor transaksi baru dan tambahan *hashing* dari catatan sebelumnya serta hasil *hashing* itu sendiri. Setelah data transaksi berhasil dicatatkan pada *database server* gerai maka program akan mencetak pesan transaksi berhasil.

### Algoritme 3 : Fungsi Server Gerai

```

1  import socket
2  import pymysql
3  from sha1 import GenerateDigest
4  from datetime import datetime
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7
8  sock.connect( ('192.168.56.101', 7500) )
9
10 while True :
11     db = pymysql.connect("localhost","root","br89q","transaksi" )
12     cursor = db.cursor()
13
14     kode = raw_input("Masukan Kode Bayar : ")
15     sql = "SELECT * FROM pemesanan WHERE kodebayar ='%s';" %
kode.strip()
16     cursor.execute(sql)
17     a=cursor.fetchall()
18     kodebayar=""
19     gerai="Gerai-01"
20     for row in a :
21         nomor,nama,id,tipe,tempatduduk,nomortiket,kereta,kelas,b
erangkat,tiba,statusbayar,harga,kodebayar=(row[0]),(row[1]),(row
[2]),(row[3]),(row[4]),(row[5]),(row[6]),(row[7]),(row[8]),(row[
9]),(row[10]),(row[11]),(row[12])
22         if kode == kodebayar :
23             print ""
24             print "Kode Bayar Sesuai"
25             print ""
26             print ("Pembayaran Atas Nama : "+ nama)
27             print ("Jumlah Pembayaran : "+ harga)
28             print ""
29             konfirmasi = raw_input("Lanjutkan Transaksi ? (y/t) : ")
30             print ""
31             if konfirmasi == "y":
32                 waktu = str(datetime.now())
33                 data
34                 =
(nama+id+tipe+tempatduduk+nomortiket+kereta+kelas+berangkat+tiba
+statusbayar+harga+kodebayar+waktu+gerai)
35                 digest =GenerateDigest(data)
36                 sock.send(digest)
37                 data = sock.recv(100)
38                 print data
39                 print""
40                 data = nama+ "$?!" +id+ "$?!" +tipe+ "$?!"
+tempatduduk+ "$?!" +nomortiket+ "$?!" +kereta+ "$?!" +kelas+
"$?!" +berangkat+ "$?!" +tiba+ "$?!" +statusbayar+ "$?!" +harga+
"$?!" +kodebayar+ "$?!" +waktu+ "$?!" +gerai
41                 sock.send(data)
42                 data = sock.recv(100)
43                 print data
44                 sql = "UPDATE pemesanan SET statusbayar='Sudah
Dibayar' WHERE nomor= (%s);" % nomor.strip()
45                 cursor.execute(sql)

```

```

46         db.commit()
47
48         sql = "SELECT max(nomor) FROM pembayaran;"
49         cursor.execute(sql)
50         b=cursor.fetchall()
51         for row in b :
52             nomor = (row[0])
53
54         if nomor != None:
55             sql = "SELECT * FROM pembayaran WHERE nomor=%s;"
56             cursor.execute(sql,(nomor))
57             b=cursor.fetchall()
58             for row in b :
59                 hashsebelum = (row[16])
60                 nomor= str(int(nomor)+1)
61
62             elif nomor == None:
63                 nomor="1"
64                 hashsebelum = "0"
65
66             hashsekarang= GenerateDigest(nomor+nama+id+tipe+
tempatduduk+nomortiket+kereta+kelas+berangkat+tiba+statusbayar+h
arga+kodebayar+waktu+gerai+hashsebelum)
67             sql = "INSERT INTO pembayaran (nomor, nama, id, tipe,
tempatduduk, nomortiket, kereta, kelas, berangkat, tiba,
statusbayar, harga, kodebayar, waktubayar, gerai, hashsebelum,
hash) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
68             cursor.execute(sql,((nomor,nama,id,tipe,tempatduduk,
nomortiket,kereta,kelas,berangkat,tiba,"Sudah
Dibayar",harga,kodebayar,waktu,gerai,hashsebelum,hashsekarang)))
69             db.commit()
70
71             elif konfirmasi == "t":
72                 print "Transaksi Dibatalkan"
73             else :
74                 print "Pilihan Salah, Transaksi Dibatalkan"
75         else :
76             print "Kode Bayar Salah"
77
78         print""
79         print""
80         db.close()

```

## BAB 6 PENGUJIAN DAN PEMBAHASAN

Pada bab ini penulis akan melakukan pengujian terhadap sistem yang telah dirancang dan dilakukan implementasi sebelumnya, penulis akan melakukan tiga pengujian, pengujian pertama ialah pengujian *test vector*, yang kedua ialah pengujian fungsionalitas, dan yang ketiga ialah pengujian *Quality of Services*.

### 6.1 Pengujian Test Vector

Pengujian *Test Vector* bertujuan untuk memastikan sub-program *SHA-1* berjalan dengan baik dan benar serta menghasilkan keluaran sesuai dengan hasil perhitungan pembuat algoritme, sebagai pembanding, penulis menggunakan *Test Vector* resmi dari National Institute of Standards and Technology (*NIST*) yang telah penulis lampirkan pada halaman lampiran.

Dalam pengujian ini penulis akan menjalankan program dengan masukan yang sama seperti *test vector*, kemudian akan membandingkan hasil perhitungan program dengan *test vector* namun sebelum program dijalankan penulis melakukan modifikasi pada kode program *SHA-1* agar program dapat mencetak hasil perhitungan tiap langkah, berikut adalah kode program yang telah penulis modifikasi:

Algoritme 4 : Fungsi SHA-1 Modifikasi	
1	def GenerateDigest(message):
2	
3	H0 = 0x67452301
4	H1 = 0xEFCDAB89
5	H2 = 0x98BADCFE
6	H3 = 0x10325476
7	H4 = 0xC3D2E1F0
8	
9	#Ubah Message kedalam bentuk biner
10	biner = ""
11	biner = biner + (''.join("{0:08b}".format(ord(x), 'b') for x
	in message))
12	
13	#Tambah 1 dibelakang pesan yang telah diubah kedalam biner
14	padded = biner+"1"
15	
16	#Tambah angka 0
17	While True
18	padded=padded+"0"
19	if (len(padded)%512 == 448%512):
20	break
21	
22	#Tambah 64bit pesan asli
23	padded=padded+'{0:064b}'.format(len(biner))
24	
25	#Function

```

26 def funct(x, y, z, i):
27     if 0 <= i <= 19:
28         f = (x & y) ^ ((~x) & z)
29     elif 20 <= i <= 39:
30         f = x ^ y ^ z
31     elif 40 <= i <= 59:
32         f = (x & y) ^ (x & z) ^ (y & z)
33     elif 60 <= i <= 79:
34         f = x ^ y ^ z
35     return f
36
37     #Constants
38 def K(i):
39     if 0 <= i <= 19:
40         k = 0x5A827999
41     elif 20 <= i <= 39:
42         k = 0x6ED9EBA1
43     elif 40 <= i <= 59:
44         k = 0x8F1BBCDC
45     elif 60 <= i <= 79:
46         k = 0xCA62C1D6
47     return k
48
49     #Sn(X)
50 def ROTL(X, n):
51     rotate = ((X << n) | (X >> (32 - n))) & 0xffffffff
52     return rotate
53
54 def split(l, n):
55     s = [l[i:i+n] for i in xrange(0, len(l), n)]
56     return s
57
58 for i in split(padded, 512):
59     w = [0]*80
60     M = split(i, 32)
61     print('')
62     print("Blok Contents :")
63     for t in range(0, 80):
64         if t <= 15 :
65             #Mt(i)
66             w[t] = int(M[t], 2)
67             print("W["+str(t)+"] = "+('%08x'%w[t]))
68
69         else :
70             #ROTL1=( tt3^tt8^tt14^tt16 )
71             w[t] = ROTL((w[t-3] ^ w[t-8] ^ w[t-14] ^ w[t-16]),
1)
72
73     #Initialize the five working variabels
74     a = H0
75     b = H1
76     c = H2
77     d = H3
78     e = H4
79
80     print('')
81

```

```

82         for i in range(0, 80):
83             T = ROTL(a,5) + funct(b,c,d,i) + e + K(i) + w[i] &
0xffffffff
84             e = d
85             d = c
86             c = ROTL(b,30)
87             b = a
88             a = T
89             print("t= "+str(i)+" : "+('%08x'%a)+" "+('%08x'%b)+"
"+('%08x'%c)+" "+('%08x'%d)+" "+('%08x'%e))
90
91             h0,h1,h2,h3,h4=H0,H1,H2,H3,H4
92
93             H0 = a + H0 & 0xffffffff
94             H1 = b + H1 & 0xffffffff
95             H2 = c + H2 & 0xffffffff
96             H3 = d + H3 & 0xffffffff
97             H4 = e + H4 & 0xffffffff
98
99             Digest =
'%08x'%H0+'%08x'%H1+'%08x'%H2+'%08x'%H3+'%08x'%H4
100
101             print('')
102             print("H[0] = "+('%08x'%h0)+" + "+('%08x'%a)+" =
"+('%08x'%H0))
103             print("H[1] = "+('%08x'%h1)+" + "+('%08x'%b)+" =
"+('%08x'%H1))
104             print("H[2] = "+('%08x'%h2)+" + "+('%08x'%c)+" =
"+('%08x'%H2))
105             print("H[3] = "+('%08x'%h3)+" + "+('%08x'%d)+" =
"+('%08x'%H3))
106             print("H[4] = "+('%08x'%h4)+" + "+('%08x'%e)+" =
"+('%08x'%H4))
107
108             return Digest
109
110 print("Message Digest is : "+(GenerateDigest
("abcbcbcdcedefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq ")))

```

### 6.1.1 Test Vector Dengan Panjang Message Satu Blok

Dari dua *test vector* yang disediakan *NIST* penulis akan mencoba *test vector* dengan panjang *message* satu blok dahulu untuk dicocokkan dengan hasil perhitungan program. *Test vector* memiliki masukan "abc" dan akan menghasilkan *digest* "A9993E364706816ABA3E25717850C26C9CD0D89D", nantinya program akan diberi masukan yang sama dengan *test vector* untuk dilihat apakah keluaran *digest* serta langkah penyelesaiannya sama atau tidak dengan *test vector*.



```

root@agung-Lenovo-Z50-75: /home/agung/koding/FIX
root@agung-Lenovo-Z50-75: /home/agung/koding/FIX# python testvector.py

Block Contents :
W[0] = 61626380
W[1] = 00000000
W[2] = 00000000
W[3] = 00000000
W[4] = 00000000
W[5] = 00000000
W[6] = 00000000
W[7] = 00000000
W[8] = 00000000
W[9] = 00000000
W[10] = 00000000
W[11] = 00000000
W[12] = 00000000
W[13] = 00000000
W[14] = 00000000
W[15] = 00000018

t= 0 : 0116fc33 67452301 7bf36ae2 98badcfe 10325476
t= 1 : 8990536d 0116fc33 59d148c0 7bf36ae2 98badcfe
t= 2 : a1390f08 8990536d c045bf0c 59d148c0 7bf36ae2
t= 3 : cdd8e11b a1390f08 626414db c045bf0c 59d148c0
t= 4 : cfd499de cdd8e11b 284e43c2 626414db c045bf0c
t= 5 : 3fc7ca40 cfd499de f3763846 284e43c2 626414db
t= 6 : 993e30c1 3fc7ca40 b3f52677 f3763846 284e43c2
t= 7 : 9e8c07d4 993e30c1 0ff1f290 b3f52677 f3763846
t= 8 : 4b6ae328 9e8c07d4 664f8c30 0ff1f290 b3f52677
t= 9 : 8351f929 4b6ae328 27a301f5 664f8c30 0ff1f290
t= 10 : fbda9e89 8351f929 12dab8ca 27a301f5 664f8c30
t= 11 : 63188fe4 fbda9e89 60d47e4a 12dab8ca 27a301f5
t= 12 : 4607b664 63188fe4 7ef6a7a2 60d47e4a 12dab8ca
t= 13 : 9128f695 4607b664 18c623f9 7ef6a7a2 60d47e4a
t= 14 : 196bee77 9128f695 1181ed99 18c623f9 7ef6a7a2
t= 15 : 20bdd62f 196bee77 644a3da5 1181ed99 18c623f9
t= 16 : 4e925823 20bdd62f c65afb9d 644a3da5 1181ed99
t= 17 : 82aa6728 4e925823 c82f758b c65afb9d 644a3da5
t= 18 : dc64901d 82aa6728 d3a49608 c82f758b c65afb9d
t= 19 : fd9e1d7d dc64901d 20aa99ca d3a49608 c82f758b
t= 20 : 1a37b0ca fd9e1d7d 77192407 20aa99ca d3a49608
t= 21 : 33a23bfc 1a37b0ca 7f67875f 77192407 20aa99ca
t= 22 : 21283486 33a23bfc 868dec32 7f67875f 77192407
t= 23 : d541f12d 21283486 0ce88eff 868dec32 7f67875f
t= 24 : c7567dc6 d541f12d 884a0d21 0ce88eff 868dec32
t= 25 : 48413ba4 c7567dc6 75507c4b 884a0d21 0ce88eff
t= 26 : be35fbd5 48413ba4 b1d59f71 75507c4b 884a0d21
t= 27 : 4aa84d97 be35fbd5 12104ee9 b1d59f71 75507c4b
t= 28 : 8370b52e 4aa84d97 6f8d7ef5 12104ee9 b1d59f71
t= 29 : c5fbaf5d 8370b52e d2aa1365 6f8d7ef5 12104ee9
t= 30 : 1267b407 c5fbaf5d a0dc2d4b d2aa1365 6f8d7ef5
t= 31 : 3b845d33 1267b407 717eebd7 a0dc2d4b d2aa1365
t= 32 : 046faa0a 3b845d33 c499ed01 717eebd7 a0dc2d4b
t= 33 : 2c0ebc11 046faa0a cee1174c c499ed01 717eebd7
t= 34 : 21796ad4 2c0ebc11 811bea82 cee1174c c499ed01
t= 35 : dcbbb0cb 21796ad4 4b03af04 811bea82 cee1174c
t= 36 : 0f511fd8 dcbbb0cb 085e5ab5 4b03af04 811bea82
t= 37 : dc63973f 0f511fd8 f72eec32 085e5ab5 4b03af04
t= 38 : 4c986405 dc63973f 03d447f6 f72eec32 085e5ab5
t= 39 : 32de1cba 4c986405 f718e5cf 03d447f6 f72eec32
t= 40 : fc87dedf 32de1cba 53261901 f718e5cf 03d447f6
t= 41 : 970a0d5c fc87dedf 8cb7872e 53261901 f718e5cf
t= 42 : 7f193dc5 970a0d5c ff21f7b7 8cb7872e 53261901
t= 43 : ee1b1aaf 7f193dc5 25c28357 ff21f7b7 8cb7872e
t= 44 : 40f28e09 ee1b1aaf 5fc64f71 25c28357 ff21f7b7
t= 45 : 1c51e1f2 40f28e09 fb86c6ab 5fc64f71 25c28357
t= 46 : a01b846c 1c51e1f2 503ca382 fb86c6ab 5fc64f71
t= 47 : bead02ca a01b846c 8714787c 503ca382 fb86c6ab
t= 48 : baf39337 bead02ca 2806e11b 8714787c 503ca382
t= 49 : 120731c5 baf39337 afab40b2 2806e11b 8714787c
t= 50 : 641db2ce 120731c5 eebce4cd afab40b2 2806e11b

```

Gambar 6.1 Hasil Perhitungan Satu Blok Bagian 1



```

t= 51 : 3847ad66 641db2ce 4481cc71 eebce4cd afab40b2
t= 52 : e490436d 3847ad66 99076cb3 4481cc71 eebce4cd
t= 53 : 27e9f1d8 e490436d 8e11eb59 99076cb3 4481cc71
t= 54 : 7b71f76d 27e9f1d8 792410db 8e11eb59 99076cb3
t= 55 : 5e6456af 7b71f76d 09fa7c76 792410db 8e11eb59
t= 56 : c846093f 5e6456af 5edc7ddb 09fa7c76 792410db
t= 57 : d262ff50 c846093f d79915ab 5edc7ddb 09fa7c76
t= 58 : 09d785fd d262ff50 f211824f d79915ab 5edc7ddb
t= 59 : 3f52de5a 09d785fd 3498bfd4 f211824f d79915ab
t= 60 : d756c147 3f52de5a 4275e17f 3498bfd4 f211824f
t= 61 : 548c9cb2 d756c147 8fd4b796 4275e17f 3498bfd4
t= 62 : b66c026b 548c9cb2 f5d5b051 8fd4b796 4275e17f
t= 63 : 6b61c9e1 b66c026b 9523272c f5d5b051 8fd4b796
t= 64 : 19dfa7ac 6b61c9e1 ed9b0082 9523272c f5d5b051
t= 65 : 101655f9 19dfa7ac 5ad87278 ed9b0082 9523272c
t= 66 : 0c3df2b4 101655f9 0677e9eb 5ad87278 ed9b0082
t= 67 : 78dd4d2b 0c3df2b4 4405957e 0677e9eb 5ad87278
t= 68 : 497093c0 78dd4d2b 030f7cad 4405957e 0677e9eb
t= 69 : 3f2588c2 497093c0 de37534a 030f7cad 4405957e
t= 70 : c199f8c7 3f2588c2 125c24f0 de37534a 030f7cad
t= 71 : 39859de7 c199f8c7 8fc96230 125c24f0 de37534a
t= 72 : edb42de4 39859de7 f0667e31 8fc96230 125c24f0
t= 73 : 11793f6f edb42de4 ce616779 f0667e31 8fc96230
t= 74 : 5ee76897 11793f6f 3b6d0b79 ce616779 f0667e31
t= 75 : 63f7dab7 5ee76897 c45e4fdb 3b6d0b79 ce616779
t= 76 : a079b7d9 63f7dab7 d7b9da25 c45e4fdb 3b6d0b79
t= 77 : 860d21cc a079b7d9 d8fdf6ad d7b9da25 c45e4fdb
t= 78 : 5738d5e1 860d21cc 681e6df6 d8fdf6ad d7b9da25
t= 79 : 42541b35 5738d5e1 21834873 681e6df6 d8fdf6ad

H[0] = 67452301 + 42541b35 = a9993e36
H[1] = efcdab89 + 5738d5e1 = 4706816a
H[2] = 98badcfe + 21834873 = ba3e2571
H[3] = 10325476 + 681e6df6 = 7850c26c
H[4] = c3d2e1f6 + d8fdf6ad = 9cd0d89d
Message Digest is : a9993e364706816aba3e25717850c26c9cd0d89d
root@agung-Lenovo-Z50-75:/home/agung/koding/FIX#

```

**Gambar 6.2 Hasil Perhitungan Satu Blok Bagian 2**

Setelah melakukan percobaan membandingkan hasil perhitungan program dengan *test vector* dapat dilihat hasil dari perhitungan sama dengan *test vector* dan dapat disimpulkan bahwa program telah sesuai dan dapat digunakan sebagai modul penghitung *SHA-1* pada sistem yang telah dibuat seperti yang dapat dilihat pada Tabel 6.1.

objek	Masukan	Keluaran
<i>Test Vector</i>	abc	A9993E364706816ABA3E25717850C26C9CD0D89D
Hasil Keluaran	abc	A9993E364706816ABA3E25717850C26C9CD0D89D

**Tabel 6.1 Hasil Pengujian Satu Blok**

### 6.1.2 Test Vector Dengan Panjang Message Dua Blok

Setelah sebelumnya menguji dengan menggunakan *test vector* yang memiliki panjang *message* satu blok, kali ini penulis akan menguji dengan menggunakan *test vector* yang memiliki panjang *message* dua blok untuk cocokan dengan hasil perhitungan program. *Test vector* memiliki masukan "abcbcbcdcedfdefgefghfghighijhijkijklklmklmnlmnomnopnopq" dan akan menghasilkan *digest* "84983E441C3BD26EBAE4AA1F95129E5E54670F1", nantinya program akan diberi masukan yang sama dengan *test vector* untuk dilihat apakah keluaran *digest* serta langkah penyelesaiannya sama atau tidak dengan *test vector*.

```
agung@agung-Lenovo-Z50-75:~$ python koding/testvector.py
```

```
Block Contents :
```

```
W[0] = 61626364
W[1] = 62636465
W[2] = 63646566
W[3] = 64656667
W[4] = 65666768
W[5] = 66676869
W[6] = 6768696a
W[7] = 68696a6b
W[8] = 696a6b6c
W[9] = 6a6b6c6d
W[10] = 6b6c6d6e
W[11] = 6c6d6e6f
W[12] = 6d6e6f70
W[13] = 6e6f7071
W[14] = 80000000
W[15] = 00000000
```

```
t= 0 : 0116fc17 67452301 7bf36ae2 98badcfe 10325476
t= 1 : ebf3b452 0116fc17 59d148c0 7bf36ae2 98badcfe
t= 2 : 5109913a ebf3b452 c045bf05 59d148c0 7bf36ae2
t= 3 : 2c4f6eac 5109913a bafced14 c045bf05 59d148c0
t= 4 : 33f4ae5b 2c4f6eac 9442644e bafced14 c045bf05
t= 5 : 96b85189 33f4ae5b 0b13dbab 9442644e bafced14
t= 6 : db04cb58 96b85189 ccf2b96 0b13dbab 9442644e
t= 7 : 45833f0f db04cb58 65ae1462 ccf2b96 0b13dbab
t= 8 : c565c35e 45833f0f 36c132d6 65ae1462 ccf2b96
t= 9 : 6350afda c565c35e d160cfc3 36c132d6 65ae1462
t= 10 : 8993ea77 6350afda b15970d7 d160cfc3 36c132d6
t= 11 : e19ecaa2 8993ea77 98d42bf6 b15970d7 d160cfc3
t= 12 : 8603481e e19ecaa2 e264fa9d 98d42bf6 b15970d7
t= 13 : 32f94a85 8603481e b867b2a8 e264fa9d 98d42bf6
t= 14 : b2e7a8be 32f94a85 a180d207 b867b2a8 e264fa9d
t= 15 : 42637e39 b2e7a8be 4cbe52a1 a180d207 b867b2a8
t= 16 : 6b068048 42637e39 acb9ea2f 4cbe52a1 a180d207
t= 17 : 426b9c35 6b068048 5098df8e acb9ea2f 4cbe52a1
t= 18 : 944b1bd1 426b9c35 1ac1a012 5098df8e acb9ea2f
t= 19 : 6c445652 944b1bd1 509ae70d 1ac1a012 5098df8e
t= 20 : 95836da5 6c445652 6512c6f4 509ae70d 1ac1a012
t= 21 : 09511177 95836da5 9b111594 6512c6f4 509ae70d
t= 22 : e2b92dc4 09511177 6560db69 9b111594 6512c6f4
t= 23 : fd224575 e2b92dc4 c254445d 6560db69 9b111594
t= 24 : eeb82d9a fd224575 38ae4b71 c254445d 6560db69
t= 25 : 5a142c1a eeb82d9a 7f48915d 38ae4b71 c254445d
t= 26 : 2972f7c7 5a142c1a bbae0b66 7f48915d 38ae4b71
t= 27 : d526a644 2972f7c7 96850b06 bbae0b66 7f48915d
t= 28 : e1122421 d526a644 ca5cbdf1 96850b06 bbae0b66
t= 29 : 05b457b2 e1122421 3549a991 ca5cbdf1 96850b06
t= 30 : a9c84bec 05b457b2 78448908 3549a991 ca5cbdf1
t= 31 : 52e31f60 a9c84bec 816d15ec 78448908 3549a991
t= 32 : 5af3242c 52e31f60 2a7212fb 816d15ec 78448908
t= 33 : 31c756a9 5af3242c 14b8c7d8 2a7212fb 816d15ec
```

Gambar 6.3 Hasil Perhitungan Dua Blok Bagian 1



```

t= 34 : e9ac987c 31c756a9 16bcc90b 14b8c7d8 2a7212fb
t= 35 : ab7c32ee e9ac987c 4c71d5aa 16bcc90b 14b8c7d8
t= 36 : 5933fc99 ab7c32ee 3a6b261f 4c71d5aa 16bcc90b
t= 37 : 43f87ae9 5933fc99 aadf0cbb 3a6b261f 4c71d5aa
t= 38 : 24957f22 43f87ae9 564cff26 aadf0cbb 3a6b261f
t= 39 : adeb7478 24957f22 50fe1eba 564cff26 aadf0cbb
t= 40 : d70e5010 adeb7478 89255fc8 50fe1eba 564cff26
t= 41 : 79bcfb08 d70e5010 2b7add1e 89255fc8 50fe1eba
t= 42 : f9bcb8de 79bcfb08 35c39404 2b7add1e 89255fc8
t= 43 : 633e9561 f9bcb8de 1e6f3ec2 35c39404 2b7add1e
t= 44 : 98c1ea64 633e9561 be6f2e37 1e6f3ec2 35c39404
t= 45 : c6ea241e 98c1ea64 58cfa558 be6f2e37 1e6f3ec2
t= 46 : a2ad4f02 c6ea241e 26307a99 58cfa558 be6f2e37
t= 47 : c8a69090 a2ad4f02 b1ba8907 26307a99 58cfa558
t= 48 : 88341600 c8a69090 a8ab53c0 b1ba8907 26307a99
t= 49 : 7e846f58 88341600 3229a424 a8ab53c0 b1ba8907
t= 50 : 86e358ba 7e846f58 220d0580 3229a424 a8ab53c0
t= 51 : 8d2e76c8 86e358ba 1fa11bd6 220d0580 3229a424
t= 52 : ce892e10 8d2e76c8 a1b8d62e 1fa11bd6 220d0580
t= 53 : edea95b1 ce892e10 234b9db2 a1b8d62e 1fa11bd6
t= 54 : 36d1230a edea95b1 33a24b84 234b9db2 a1b8d62e
t= 55 : 776c3910 36d1230a 7b7aa56c 33a24b84 234b9db2
t= 56 : a681b723 776c3910 8db448c2 7b7aa56c 33a24b84
t= 57 : ac0a794f a681b723 1ddb0e44 8db448c2 7b7aa56c
t= 58 : f03d3782 ac0a794f e9a06dc8 1ddb0e44 8db448c2
t= 59 : 9ef775c3 f03d3782 eb029e53 e9a06dc8 1ddb0e44
t= 60 : 36254b13 9ef775c3 bc0f4de0 eb029e53 e9a06dc8
t= 61 : 4080d4dc 36254b13 e7bddd70 bc0f4de0 eb029e53
t= 62 : 2bfaf7a8 4080d4dc cd8952c4 e7bddd70 bc0f4de0
t= 63 : 513f9ca0 2bfaf7a8 10203537 cd8952c4 e7bddd70
t= 64 : e5895c81 513f9ca0 0afebdea 10203537 cd8952c4
t= 65 : 1037d2d5 e5895c81 144fe728 0afebdea 10203537
t= 66 : 14a82da9 1037d2d5 79625720 144fe728 0afebdea
t= 67 : 6d17c9fd 14a82da9 440df4b5 79625720 144fe728
t= 68 : 2c7b07bd 6d17c9fd 452a0b6a 440df4b5 79625720
t= 69 : fdf6efff 2c7b07bd 5b45f27f 452a0b6a 440df4b5
t= 70 : 112b96e3 fdf6efff 4b1ec1ef 5b45f27f 452a0b6a
t= 71 : 84065712 112b96e3 ff7dbbff 4b1ec1ef 5b45f27f
t= 72 : ab89fb71 84065712 c44ae5b8 ff7dbbff 4b1ec1ef
t= 73 : c5210e35 ab89fb71 a10195c4 c44ae5b8 ff7dbbff
t= 74 : 352d9f4b c5210e35 6ae27edc a10195c4 c44ae5b8
t= 75 : 1a0e0e0a 352d9f4b 7148438d 6ae27edc a10195c4
t= 76 : d0d47349 1a0e0e0a cd4b67d2 7148438d 6ae27edc
t= 77 : ad38620d d0d47349 86838382 cd4b67d2 7148438d
t= 78 : d3ad7c25 ad38620d 74351cd2 86838382 cd4b67d2
t= 79 : 8ce34517 d3ad7c25 6b4e1883 74351cd2 86838382

```

```

H[0] = 67452301 + 8ce34517 = f4286818
H[1] = efcdab89 + d3ad7c25 = c37b27ae
H[2] = 98badcfe + 6b4e1883 = 0408f581
H[3] = 10325476 + 74351cd2 = 84677148
H[4] = c3d2e1f0 + 86838382 = 4a566572

```

Gambar 6.4 Hasil Perhitungan Dua Blok Bagian 2

```

Block Contents :
W[0] = 00000000
W[1] = 00000000
W[2] = 00000000
W[3] = 00000000
W[4] = 00000000
W[5] = 00000000
W[6] = 00000000
W[7] = 00000000
W[8] = 00000000
W[9] = 00000000
W[10] = 00000000
W[11] = 00000000
W[12] = 00000000
W[13] = 00000000
W[14] = 00000000
W[15] = 000001c0

t= 0 : 2df257e9 f4286818 b0dec9eb 0408f581 84677148
t= 1 : 4d3dc58f 2df257e9 3d0a1a06 b0dec9eb 0408f581
t= 2 : c352bb05 4d3dc58f 4b7c95fa 3d0a1a06 b0dec9eb
t= 3 : eef743c6 c352bb05 d34f7163 4b7c95fa 3d0a1a06
t= 4 : 41e34277 eef743c6 70d4aec1 d34f7163 4b7c95fa
t= 5 : 5443915c 41e34277 bbbdd0f1 70d4aec1 d34f7163
t= 6 : e7fa0377 5443915c d078d09d bbbdd0f1 70d4aec1
t= 7 : c6946813 e7fa0377 1510e457 d078d09d bbbdd0f1
t= 8 : fdde1de1 c6946813 f9fe80dd 1510e457 d078d09d
t= 9 : b8538aca fdde1de1 f1a51a04 f9fe80dd 1510e457
t= 10 : 6ba94f63 b8538aca 7f778778 f1a51a04 f9fe80dd
t= 11 : 43a2792f 6ba94f63 ae14e2b2 7f778778 f1a51a04
t= 12 : fecd7bbf 43a2792f daea53d8 ae14e2b2 7f778778
t= 13 : a2604ca8 fecd7bbf d0e89e4b daea53d8 ae14e2b2
t= 14 : 258b0baa a2604ca8 ffb35eef d0e89e4b daea53d8
t= 15 : d9772360 258b0baa 2898132a ffb35eef d0e89e4b
t= 16 : 5507db6e d9772360 8962c2ea 2898132a ffb35eef
t= 17 : a51b58bc 5507db6e 365dc8d8 8962c2ea 2898132a
t= 18 : c2eb709f a51b58bc 9541f6db 365dc8d8 8962c2ea
t= 19 : d8992153 c2eb709f 2946d62f 9541f6db 365dc8d8
t= 20 : 37482f5f d8992153 f0badc27 2946d62f 9541f6db
t= 21 : ee8700bd 37482f5f f6264854 f0badc27 2946d62f
t= 22 : 9ad594b9 ee8700bd cdd20bd7 f6264854 f0badc27
t= 23 : 8fbaa5b9 9ad594b9 7ba1c02f cdd20bd7 f6264854
t= 24 : 88fb5867 8fbaa5b9 66b5652e 7ba1c02f cdd20bd7
t= 25 : eec50521 88fb5867 63eea96e 66b5652e 7ba1c02f
t= 26 : 50bce434 eec50521 e23ed619 63eea96e 66b5652e
t= 27 : 5c416daf 50bce434 7bb14148 e23ed619 63eea96e
t= 28 : 2429be5f 5c416daf 142f390d 7bb14148 e23ed619
t= 29 : 0a2fb108 2429be5f d7105b6b 142f390d 7bb14148
t= 30 : 17986223 0a2fb108 c90a6f97 d7105b6b 142f390d
t= 31 : 8a4af384 17986223 028bec42 c90a6f97 d7105b6b
t= 32 : 6b629993 8a4af384 c5e61888 028bec42 c90a6f97
t= 33 : f15f04f3 6b629993 2292bce1 c5e61888 028bec42
t= 34 : 295cc25b f15f04f3 dad8a664 2292bce1 c5e61888

```

Gambar 6.5 Hasil Perhitungan Dua Blok Bagian 3



```

t= 35 : 696da404 295cc25b fc57c13c dad8a664 2292bce1
t= 36 : cef5ae12 696da404 ca573096 fc57c13c dad8a664
t= 37 : 87d5b80c cef5ae12 1a5b6901 ca573096 fc57c13c
t= 38 : 84e2a5f2 87d5b80c b3bd6b84 1a5b6901 ca573096
t= 39 : 03bb6310 84e2a5f2 21f56e03 b3bd6b84 1a5b6901
t= 40 : c2d8f75f 03bb6310 a138a97c 21f56e03 b3bd6b84
t= 41 : bfb25768 c2d8f75f 00eed8c4 a138a97c 21f56e03
t= 42 : 28589152 bfb25768 f0b63dd7 00eed8c4 a138a97c
t= 43 : ec1d3d61 28589152 2fec95da f0b63dd7 00eed8c4
t= 44 : 3caed7af ec1d3d61 8a162454 2fec95da f0b63dd7
t= 45 : c3d033ea 3caed7af 7b074f58 8a162454 2fec95da
t= 46 : 7316056a c3d033ea cf2bb5eb 7b074f58 8a162454
t= 47 : 46f93b68 7316056a b0f40cfa cf2bb5eb 7b074f58
t= 48 : dc8e7f26 46f93b68 9cc5815a b0f40cfa cf2bb5eb
t= 49 : 850d411c dc8e7f26 11be4eda 9cc5815a b0f40cfa
t= 50 : 7e4672c0 850d411c b7239fc9 11be4eda 9cc5815a
t= 51 : 89fbd41d 7e4672c0 21435047 b7239fc9 11be4eda
t= 52 : 1797e228 89fbd41d 1f919cb0 21435047 b7239fc9
t= 53 : 431d65bc 1797e228 627ef507 1f919cb0 21435047
t= 54 : 2bdbb8cb 431d65bc 05e5f88a 627ef507 1f919cb0
t= 55 : 6da72e7f 2bdbb8cb 10c7596f 05e5f88a 627ef507
t= 56 : a8495a9b 6da72e7f caf6ee32 10c7596f 05e5f88a
t= 57 : e785655a a8495a9b db69cb9f caf6ee32 10c7596f
t= 58 : 5b086c42 e785655a ea1256a6 db69cb9f caf6ee32
t= 59 : a65818f7 5b086c42 b9e15956 ea1256a6 db69cb9f
t= 60 : 7aab101b a65818f7 96c21b10 b9e15956 ea1256a6
t= 61 : 93614c9c 7aab101b e996063d 96c21b10 b9e15956
t= 62 : f66d9bf4 93614c9c deaac406 e996063d 96c21b10
t= 63 : d504902b f66d9bf4 24d85327 deaac406 e996063d
t= 64 : 60a9da62 d504902b 3d9b66fd 24d85327 deaac406
t= 65 : 8b687819 60a9da62 f541240a 3d9b66fd 24d85327
t= 66 : 083e90c3 8b687819 982a7698 f541240a 3d9b66fd
t= 67 : f6226bbf 083e90c3 62da1e06 982a7698 f541240a
t= 68 : 76c0563b f6226bbf c20fa430 62da1e06 982a7698
t= 69 : 989dd165 76c0563b fd889aef c20fa430 62da1e06
t= 70 : 8b2c7573 989dd165 ddb0158e fd889aef c20fa430
t= 71 : ae1b8e7b 8b2c7573 66277459 ddb0158e fd889aef
t= 72 : ca1840de ae1b8e7b e2cb1d5c 66277459 ddb0158e
t= 73 : 16f3babb ca1840de eb86e39e e2cb1d5c 66277459
t= 74 : d28d83ad 16f3babb b2861037 eb86e39e e2cb1d5c
t= 75 : 6bc02dfe d28d83ad c5bceeae b2861037 eb86e39e
t= 76 : d3a6e275 6bc02dfe 74a360eb c5bceeae b2861037
t= 77 : da955482 d3a6e275 9af00b7f 74a360eb c5bceeae
t= 78 : 58c0aac0 da955482 74e9b89d 9af00b7f 74a360eb
t= 79 : 906fd62c 58c0aac0 b6a55520 74e9b89d 9af00b7f

H[0] = f4286818 + 906fd62c = 84983e44
H[1] = c37b27ae + 58c0aac0 = 1c3bd26e
H[2] = 0408f581 + b6a55520 = baae4aa1
H[3] = 84677148 + 74e9b89d = f95129e5
H[4] = 4a566572 + 9af00b7f = e54670f1
Message Digest is : 84983e441c3bd26ebaae4aa1f95129e5e54670f1
agung@agung-Lenovo-Z50-75:~$

```

**Gambar 6.6 Hasil Perhitungan Dua Blok Bagian 4**

Setelah melakukan percobaan membandingkan hasil perhitungan program dengan *test vector* dapat dilihat hasil dari perhitungan sama dengan *test vector* dan dapat disimpulkan bahwa program telah sesuai dan dapat digunakan sebagai modul penghitung *SHA-1* pada sistem yang telah dibuat dibuat seperti yang dapat dilihat pada Tabel 6.2.

objek	Masukan	Keluaran
<i>Test Vector</i>	abcdbcdecdefdefgefghfghighijhij kijkljklmklmnlmnomnopnopq	84983E441C3BD26EBAAE4AA 1F95129E5E54670F1
Hasil Keluaran	Abcdbcdecdefdefgefghfghighijhij kijkljklmklmnlmnomnopnopq	84983E441C3BD26EBAAE4AA 1F95129E5E54670F1

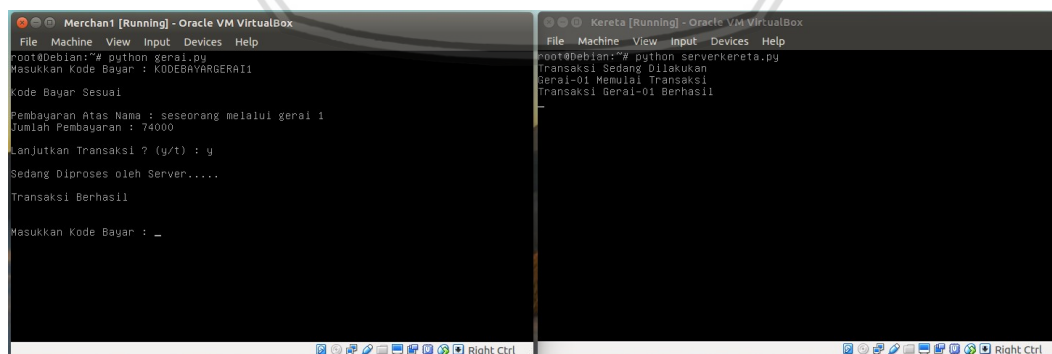
Tabel 6.2 Hasil Pengujian Dua Blok

## 6.2 Pengujian Fungsionalitas

Pengujian fungsionalitas adalah pengujian untuk mengetahui apakah sistem dapat menjalankan tugas sesuai fungsinya dengan baik dan benar atau sebaliknya, penulis melakukan pengujian fungsionalitas dengan cara melakukan transaksi dari empat *server* gerai yang berbeda kepada *server* kereta api kemudian dilihat apakah pembayaran dapat dicatatkan pada *server* kereta api dengan skema pengamanan atau sebaliknya. Percobaan akan dilakukan dengan lima mesin *virtual* dengan masing-masing empat mesin *virtual* untuk *server* gerai dan satu mesin *virtual* untuk *server* kereta, semua mesin menggunakan sistem operasi linux Debian 8. Berikut percobaan yang penulis lakukan:

### 6.2.1 Transaksi dari gerai 1

Transaksi akan dilakukan dari *server* gerai ke *server* kereta dengan asumsi tiket telah dipesan dan telah tercatat dalam *database* pemesanan pada *server* gerai, kemudian saat transaksi *server* gerai akan meminta kode bayar dan akan dicocokkan dengan data pemesanan. Jika kode bayar ditemukan maka *server* gerai akan meminta konfirmasi untuk melanjutkan transaksi, saat transaksi dilanjutkan *server* gerai akan mengirimkan *digest* data transaksi dan data transaksi ke *server* kereta untuk disimpan dalam *database* *server* kereta dan *database* gerai sendiri dan diganti statusnya menjadi “Sudah Dibayar”, rangkaian transaksi tersebut telah penulis lakukan dan dapat dilihat pada Gambar 6.7 .



Gambar 6.7 Transaksi Dari Gerai 1

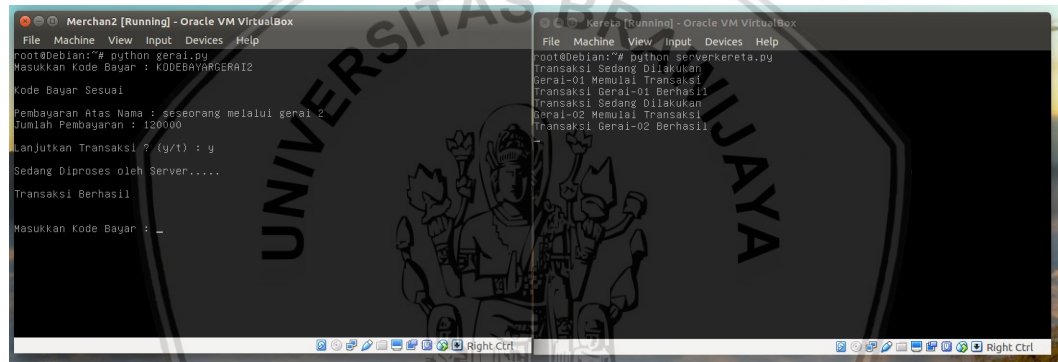
Sehingga akan menghasilkan catatan transaksi dengan tambahan *digest* data transaksi pada *server* gerai sebagai berikut:

nomor	nama	id	jenis	tempatduduk	nomortiket	kereta	kelas	berangkat	tiba	statusbayar	harga	kodebayar	waktubayar	gerai	hashsebelum	hash
1	seorang melalui gerai 1	32454345345221	DEWASA	18 C	4834532222254	LOGOWA	1K20C0M	Bangi 09.00	Lempuyangan 16.00	Belum Dibayar	74600	KODEBARANGERA1	2018-05-18 03:22:42.860913	Gerai-01	0	a77b913dad93bf63ee900aa207348b204d0874

Gambar 6.8 Database Gerai 1

### 6.2.2 Transaksi dari gerai 2

Transaksi akan dilakukan dari *server* gerai ke *server* kereta dengan asumsi tiket telah dipesan dan telah tercatat dalam *database* pemesanan pada *server* gerai, kemudian saat transaksi *server* gerai akan meminta kode bayar dan akan dicocokkan dengan data pemesanan. Jika kode bayar ditemukan maka *server* gerai akan meminta konfirmasi untuk melanjutkan transaksi, saat transaksi dilanjutkan *server* gerai akan mengirimkan *digest* data transaksi dan data transaksi ke *server* kereta untuk disimpan dalam *database server* kereta dan *database* gerai sendiri dan diganti statusnya menjadi “Sudah Dibayar”, rangkaian transaksi tersebut telah penulis lakukan dan dapat dilihat pada Gambar 6.9 .



Gambar 6.9 Transaksi Dari Gerai 2

Sehingga akan menghasilkan catatan transaksi dengan tambahan *digest* data transaksi pada *server* gerai sebagai berikut:

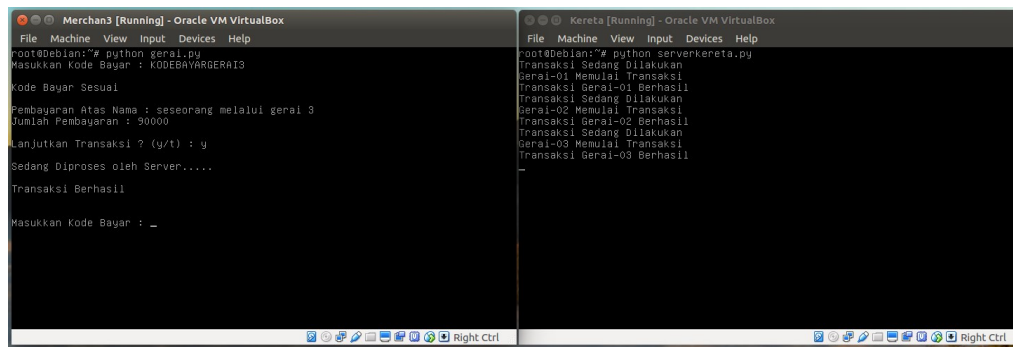
nomor	nama	id	jenis	tempatduduk	nomortiket	kereta	kelas	berangkat	tiba	statusbayar	harga	kodebayar	waktubayar	gerai	hashsebelum	hash
1	seorang melalui gerai 2	2654654634545222	DEWASA	17 A	34354342346723765	JABONGA	1K20C0M	Gubeng 21.00	Ngali 04.00	Belum Dibayar	120000	KODEBARANGERA2	2018-05-18 03:25:37.874219	Gerai-02	0	cc802a19668000b4974c3434c48b065b14feb

Gambar 6.10 Database Gerai 2

### 6.2.3 Transaksi dari gerai 3

Transaksi akan dilakukan dari *server* gerai ke *server* kereta dengan asumsi tiket telah dipesan dan telah tercatat dalam *database* pemesanan pada *server* gerai, kemudian saat transaksi *server* gerai akan meminta kode bayar dan akan dicocokkan dengan data pemesanan. Jika kode bayar ditemukan maka *server* gerai akan meminta konfirmasi untuk melanjutkan transaksi, saat transaksi dilanjutkan *server* gerai akan mengirimkan *digest* data transaksi dan data transaksi ke *server* kereta untuk disimpan dalam *database server* kereta dan *database* gerai sendiri dan diganti statusnya menjadi “Sudah Dibayar”, rangkaian transaksi tersebut telah penulis lakukan dan dapat dilihat pada Gambar 6.11 .





Gambar 6.11 Transaksi Dari Gerai 3

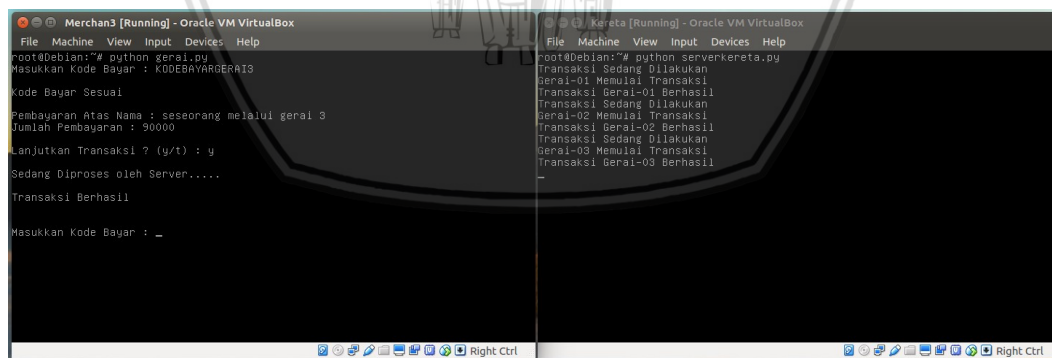
Sehingga akan menghasilkan catatan transaksi dengan tambahan *digest* data transaksi pada *server* gerai sebagai berikut:

nomor	nama	id	type	tempataduk	nomortiket	kereta	kelas	berangkat	tiba	statusbayar	harga	kodebayar	waktubayar	gerai	hashsebelum	hash
1	seseorang melalui gerai 3	43546345345222	DEWISA	21 B	564342345723765	RSUNDARA	ECINOWI	Gubeng 06.00	Karascondong 23.00	Sudah Dibayar	90000	KODEBAYARGERAI3	2018-05-18 03:27:13.508579	Gerai-03	0	87ab12d26a8b8c6d51372d18d9534ad7b67f

Gambar 6.12 Database Gerai 3

#### 6.2.4 Transaksi dari gerai 4

Transaksi akan dilakukan dari *server* gerai ke *server* kereta dengan asumsi tiket telah dipesan dan telah tercatat dalam *database* pemesanan pada *server* gerai, kemudian saat transaksi *server* gerai akan meminta kode bayar dan akan dicocokkan dengan data pemesanan. Jika kode bayar ditemukan maka *server* gerai akan meminta konfirmasi untuk melanjutkan transaksi, saat transaksi dilanjutkan *server* gerai akan mengirimkan *digest* data transaksi dan data transaksi ke *server* kereta untuk disimpan dalam *database server* kereta dan *database* gerai sendiri dan diganti statusnya menjadi “Sudah Dibayar”, rangkaian transaksi tersebut telah penulis lakukan dan dapat dilihat pada Gambar 6.13 .



Gambar 6.13 Transaksi Dari Gerai 4

Sehingga akan menghasilkan catatan transaksi dengan tambahan *digest* data transaksi pada *server* gerai sebagai berikut:

nomor	nama	id	type	tempataduk	nomortiket	kereta	kelas	berangkat	tiba	statusbayar	harga	kodebayar	waktubayar	gerai	hashsebelum	hash
1	seseorang melalui gerai 3	43546345345222	DEWISA	21 B	564342345723765	RSUNDARA	ECINOWI	Gubeng 06.00	Karascondong 23.00	Sudah Dibayar	90000	KODEBAYARGERAI3	2018-05-18 03:27:13.508579	Gerai-03	0	87ab12d26a8b8c6d51372d18d9534ad7b67f

Gambar 6.14 Database Gerai 4

## 6.2.5 Hasil Pencatatan Transaksi Server Kereta

Setelah melakukan beberapa transaksi dari empat gerai, data transaksi akan disimpan kedalam *database server* kereta sesuai urutan transaksi dan *digest* transaksi sebelumnya akan dicatatkan pada transaksi setelahnya sebagai tambahan data untuk dihitung *digest* transaksi tersebut. Catatan transaksi pada *database* kereta pada percobaan ini dapat dilihat pada Gambar 6.15 .

nomor	nama	id	type	tempat duduk	nomor tiket	kereta	kelas	berangkat	tiba	status bayar	harga	kode bayar	waktu bayar	gerai	hash sebelum	hash
1	senorang melalui gerai 1	12454343343223	DEWASA	18 C	463432323234	LOGARA	KECOWA	Bangi 09.00	Lempayangan 16.00	Sudah Dibayar	12000	KODEBARANGERAI	2018-05-18 03:22:42.809013	Gerai-01	0	a7798134a0f30a3c9e90aa2073a9-20a60874
2	senorang melalui gerai 2	20546343343223	DEWASA	11 A	3435432346723705	PAWORA	KECOWA	Gubung 23.00	Tegal 04.00	Sudah Dibayar	12000	KODEBARANGERAI	2018-05-18 03:25:57.874419	Gerai-02	a7798134a0f30a3c9e90aa2073a9-20a60874	690f181254a62081a4a012a9c3bae377930e
3	senorang melalui gerai 3	45546343343223	DEWASA	21 B	564342346723705	PAWORA	KECOWA	Gubung 08.00	Karascondong 23.00	Sudah Dibayar	90000	KODEBARANGERAI	2018-05-18 03:27:13.508579	Gerai-03	690f181254a62081a4a012a9c3bae377930e	20ac9fc5d54a22077885052aaf981729d53
4	Agung Pambudi	23432323234234	DEWASA	12 B	23423432323432	DSMO	KECOWA	Surabaya 17.00	Kali 20.00	Sudah Dibayar	10000	KODEBARANGERAI	2018-05-18 03:28:15.730242	Gerai-04	20ac9fc5d54a22077885052aaf981729d53	375a3709802400af3226a09f30a3c9e90aa2073a9-20a60874

Gambar 6.15 Database Server Kereta

## 6.3 Pengujian Validitas Data Transaksi

Pengujian validitas bertujuan untuk menguji ketepatan urutan data transaksi setiap barisnya yang telah tersimpan dalam *database* untuk memastikan integritas data transaksi sesuai tujuan dari sistem. Dalam pengujian ini penulis membuat sebuah program tambahan yang akan menghitung *digest* dari data transaksi dan *digest* data transaksi sebelumnya untuk membandingkan dengan *digest* data transaksi dan *digest* data transaksi sebelumnya yang telah tercatat dalam *database* mulai dari data baris pertama hingga data baris terakhir untuk memastikan data valid atau data tidak mengalami modifikasi.

Program ini dibuat dengan bahasa pemrograman Python, cara kerjanya pertama-tama akan melakukan *import library* PyMySQL untuk mengelola *database* dan modul *SHA-1* untuk melakukan perhitungan *digest*, kemudian inisialisasi variabel pembantu seperti *counter*, *temphash* dan *maxno* dilanjutkan dengan inisialisasi *database*. Setelah selesai inisialisasi maka program akan melihat nomor terbesar pada catatan data transaksi sebagai acuan perhitungan, program akan membandingkan perhitungan *hash* dengan catatan pada *database* dan nilai *digest* catatan *digest hash* sebelumnya yang ada pada catatan dengan nilai *digest hash* sebelumnya, jika selesai pada satu baris maka variabel *counter* akan ditambah sehingga program akan memeriksa baris selanjutnya hingga nilai variabel *counter* sama dengan banyaknya baris plus satu, hal ini menandakan bahwa semua baris telah diperiksa dan memenuhi syarat sehingga variabel *counter* mampu bertambah hingga melewati nilai maksimal baris catatan kemudian akan dicetak pesan bahwa data transaksi yang telah diperiksa valid, sebaliknya jika dalam pemeriksaan ditemukan data transaksi yang tidak memenuhi syarat maka program akan berhenti memeriksa dan mencetak pesan data transaksi tidak valid.

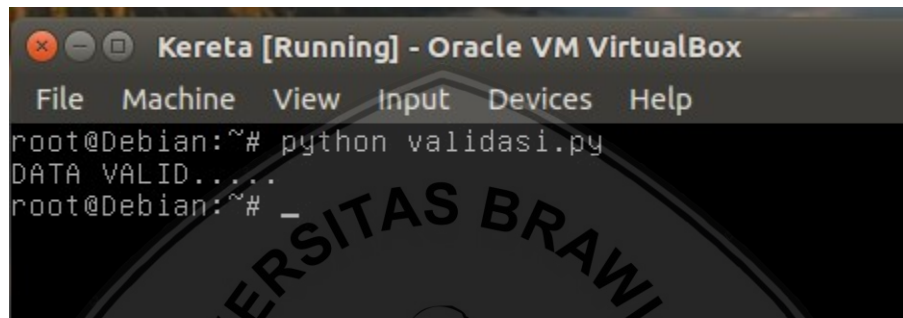
Algoritme 5 : Fungsi Validitas	
1	import pymysql
2	from sha1 import GenerateDigest
3	
4	counter=1

```

5  temphash=""
6  maxno=1
7  db = pymysql.connect("localhost","root","br89q","transaksi" )
8  cursor = db.cursor()
9
10 sql = "SELECT max(nomor) FROM pembayaran;"
11 cursor.execute(sql)
12 b=cursor.fetchall()
13 for row in b :
14     nomor = (row[0])
15
16 if nomor != None :
17     maxno=int(nomor)
18     while True:
19         sql = "SELECT * FROM pembayaran WHERE nomor = '%s';"
20         cursor.execute(sql,(counter))
21         a=cursor.fetchall()
22         for row in a :
23
24             nomor,nama,id,tipe,tempatduduk,nomortiket,kereta,kelas,berangkat
,tiba,statusbayar,harga,kodebayar,waktubayar,gerai,hashsebelum,h
ashsekarang=(row[0]),(row[1]),(row[2]),(row[3]),(row[4]),(row[5]
),(row[6]),(row[7]),(row[8]),(row[9]),(row[10]),(row[11]),(row[1
2]),(row[13]),(row[14]),(row[15]),(row[16])
25
26
27 hashtemp=GenerateDigest(nomor+nama+id+tipe+tempatduduk+nomortike
t+kereta+kelas+berangkat+tiba+statusbayar+harga+kodebayar+waktub
ayar+gerai+hashsebelum)
28
29         if (hashtemp==hashsekarang and hashsebelum == temphash
and counter != (maxno+1)):
30             temphash=hashsekarang
31             counter+=1
32
33         elif (counter==(maxno+1)):
34             print "DATA VALID....."
35
36         else:
37             print "DATA TIDAK VALID....."
38             break
39
40 elif nomor == None:
41     print "DATABASE KOSONG....."
42
43 db.close()
44

```

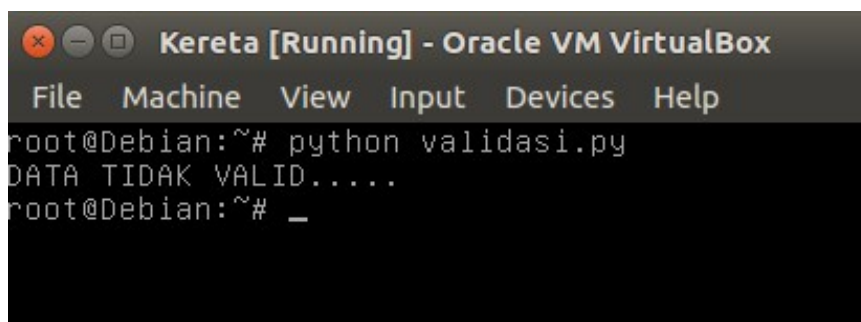
Pada pengujian fungsionalitas sebelumnya penulis melakukan percobaan terhadap fungsionalitas sistem dan menghasilkan catatan data transaksi pada *database server* kereta, maka pada pengujian kali ini penulis akan menjalankan program yang telah dibuat untuk mengoreksi catatan data transaksi pada pengujian fungsionalitas sebelumnya. Program dijalankan pada *server* kereta serta dikoneksikan dengan *database server* kereta, setelah program dijalankan didapati bahwa catatan data transaksi yang ada pada *database server* kereta valid dan integritasnya terjamin seperti yang bisa dilihat pada Gambar 6.16.



```
Kereta [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@Debian:~# python validasi.py
DATA VALID.....
root@Debian:~# _
```

Gambar 6.16 Pengujian Validitas Data Transaksi 1

Jika pada pengujian diatas dapat dibuktikan bahwa sistem akan menghasilkan catatan data yang valid, maka selanjutnya penulis akan mencoba membuktikan jika komponen data pada catatan data transaksi ada yang dirubah akan merusak susunan *hashing* sehingga integritas data tidak dapat dipercaya. Penulis akan merubah komponen data yang dihasilkan oleh percobaan sebelumnya dan akan menjalankan program validitas ulang, sehingga didapatkan hasil seperti Gambar 6.17 dan dapat disimpulkan bahwa dengan skema ini jika komponen data transaksi berubah akan menghasilkan *digest* yang berbeda sehingga akan memudahkan ketika membuktikan integritas catatan data transaksi.

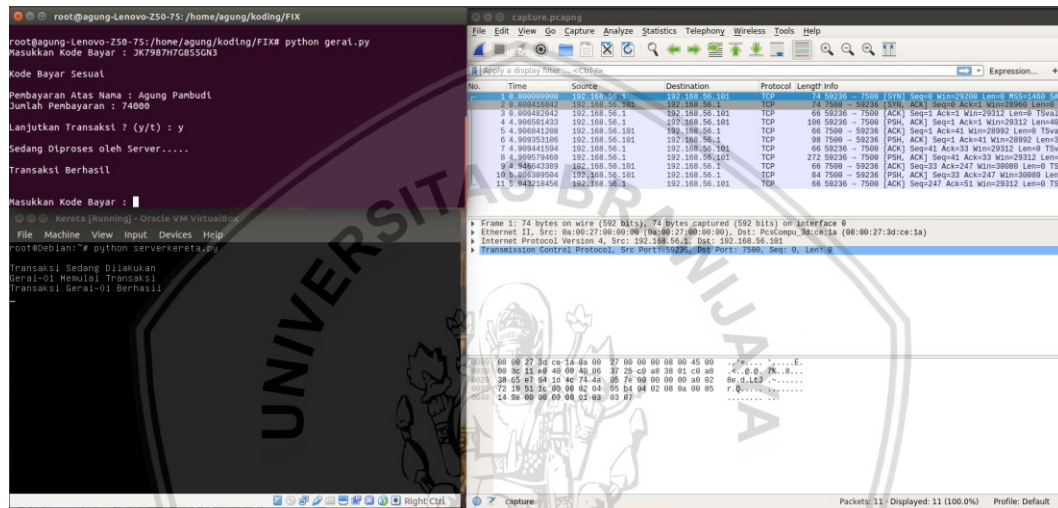


```
Kereta [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@Debian:~# python validasi.py
DATA TIDAK VALID.....
root@Debian:~# _
```

Gambar 6.17 Pengujian Validitas Data Transaksi 2

## 6.4 Pengujian *Quality of Services*

Setelah beberapa pengujian yang telah penulis lakukan dan didapatkan hasil bahwa sistem dapat berjalan dengan baik dan benar, pada pengujian ini penulis akan melakukan beberapa pengujian di antaranya berapa waktu eksekusi sistem, berapa *throughput* koneksi antara *server* gerai dan *server* kereta, adakah *packet loss* dalam transmisi data antara *server* gerai dan *server* kereta. Penulis menggunakan aplikasi Wireshark untuk mengamati lalu lintas pada jaringan yang menghubungkan antara *server* gerai dan *server* kereta seperti pada Gambar 6.18, dan hasil pengujian ini adalah sebagai berikut:



Gambar 6.18 Pengujian Validitas Data Transaksi

### 6.4.1 Waktu Eksekusi Transaksi

Waktu yang dibutuhkan dalam sekali transaksi adalah dengan menghitung waktu saat paket pertama dikirimkan sampai paket terakhir diterima, dari tangkapan aplikasi Wireshark dapat diketahui bahwa paket pertama dikirim pada pukul 00:10:49 dan paket terakhir yang merupakan balasan dari *server* kereta diterima pada pukul 00:10:54 sehingga lama waktu sekali transaksi adalah lima detik seperti yang dapat dilihat pada Gambar 6.19

Time	
First packet:	2018-05-25 00:10:49
Last packet:	2018-05-25 00:10:54
Elapsed:	00:00:05

Gambar 6.19 Waktu Eksekusi Transaksi



#### 6.4.2 Throughput

*Throughput* adalah kemampuan sebuah jaringan mentransmisikan beberapa data dalam satu waktu tertentu atau sering juga disebut sebagai *bandwidth* aktual, cara menghitungnya ialah membagi banyaknya data yang berhasil di transmisikan dengan seberapa lama data tersebut di transmisikan, terlihat dari Gambar 6.20 banyaknya data yang di transmisikan adalah 1038 *bytes* dan waktu yang dibutuhkan untuk mentransmisikan adalah 5,043 detik dari data tersebut dapat diketahui bahwa *throughput* dari transaksi antara *server* gerai dan *server* kereta adalah 205,829 *bytes/detik*.

Time span, s	5.043
Average pps	2.2
Average packet size, B	94.5
Bytes	1038
Average bytes/s	205

Gambar 6.20 *Throughput*

#### 6.4.3 Packet Loss

Sistem ini menggunakan *protocol tcp* untuk mentransmisikan data, secara teori jika koneksi tidak terputus maka tidak akan mungkin ada *packet loss*, karena sifat *protocol tcp* yang akan melakukan transmisi ulang jika ditemukan adanya *packet* yang hilang. Dengan percobaan ini penulis membuktikan bahwa dalam transmisinya sistem ini tidak menghasilkan *packet loss* seperti yang dapat dilihat pada Gambar 6.21 bahwa *packet* yang berhasil di transmisikan adalah 100%.

Measurement	Captured	Displayed
Packets	11	11 (100.0%)

Gambar 6.21 Transmisi *Packet*

## BAB 7 KESIMPULAN DAN SARAN

### 7.1 Kesimpulan

Setelah melakukan penelitian yang berjudul Perancangan Sistem Pengamanan Data Transaksi Pada *Database* Terdistribusi Menggunakan Metode *Hashing* dengan beberapa tahapannya, penulis dapat menyimpulkan beberapa hal sebagai berikut:

1. Algoritme *Secure Hash Algorithm 1 (SHA-1)* adalah algoritme untuk menjaga integritas yang pernah dijadikan standar algoritme *hashing* dan memiliki keamanan yang cukup baik karena arsitekturnya memiliki ukuran 512 blok dan menghasilkan 160 bit *digest*. Hal ini menjadikan *SHA-1* tidak mudah untuk diserang dengan metode *brute force* karena akan memerlukan waktu yang lama.
2. Dalam sistem ini penulis telah melakukan implementasi algoritme *Secure Hash Algorithm 1 (SHA-1)* dengan bahasa pemrograman yang sama dengan sub-program yang lain yaitu bahasa pemrograman Python dan dapat disimpulkan bahwa sub-program *SHA-1* dapat berjalan dengan baik dan benar sebagaimana fungsinya.
3. Dari percobaan yang telah penulis lakukan dapat disimpulkan bahwa catatan data transaksi yang telah diamankan oleh sistem ini tidak dapat dimanipulasi, karena satu catatan saling terkait dengan catatan yang lain pada *database* sehingga apabila satu catatan diubah maka akan merusak catatan yang lain, hal ini dapat diketahui menggunakan program penguji validitas data transaksi yang telah penulis buat sebelumnya.

### 7.2 Saran

Penulis menyadari masih banyak kekurangan dalam penelitian ini, di antaranya adalah tidak amannya data transaksi yang ditransmisikan dari serangan *man in middle attack* dikarenakan dalam penelitian ini penulis hanya berfokus pada integritas data serta saat ini penggunaan algoritme *hashing Secure Hash Algorithm 1 (SHA-1)* mulai digantikan dengan algoritme-algoritme yang lebih baru guna meningkatkan keamanan dan kecepatan perhitungan. Hal ini dapat dimanfaatkan sebagai topik penelitian untuk menyempurnakan penelitian yang telah penulis lakukan, penulis menyarankan penggunaan algoritme *Secure Hash Algorithm 3 (SHA-3)* sebagai pengganti algoritme *SHA-1* dan penggunaan algoritme enkripsi untuk mengamankan data transaksi pada saat ditransmisikan guna menghindari adanya serangan *man in middle attack*.



## DAFTAR PUSTAKA

- Dang, Q. H., 2015. *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication.
- Menezes, A. J., Katz, J., van Oorschot, P. C. & Vanstone, S. A., 1996. *Handbook of Applied Cryptography*. s.l.:CRC Press.
- Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*.
- Pamungkas A. A., 2006. *Implementasi Algoritma MD5, SHA1, dan RC4 untuk Sistem Kriptografi Pada Aplikasi Mobile Internet Berbasis JAVA*.
- Dewanagn, R. R., Katz, J., Thombre, D. & Sharma, K., 2015. *Implementation of Secure Hash Algorithm Using JAVA Programming*.

